Mean field optimal Core Allocation across Malleable jobs

Zhouzi Li[a], Mor Harchol-Balter[a], Benjamin Berg[b]

[a] *Computer Science Department, Carnegie Mellon University, United States*
[b] *Computer Science Department, UNC Chapel Hill, United States*

Abstract

Modern data centers and cloud computing clusters are increasingly running workloads composed of malleable jobs. A malleable job can be parallelized across any number of cores, yet the job typically exhibits diminishing marginal returns for each additional core on which it runs. This can be seen in the concavity of a job's speedup function, which describes the job's processing speed as a function of the number of cores on which it runs. Examples of malleable jobs include machine learning tasks, parallelized across multiple GPUs, and large-scale database queries, parallelized across multiple servers.

Given the prevalence of malleable jobs, theoretical works have posed the problem of how to allocate a fixed number of cores across a stream of arriving malleable jobs so as to minimize the mean response time across jobs. We refer to this as the Core Allocation to Malleable jobs (CAM) problem. Although quite a few recent papers have been written on CAM, the problem remains challenging to solve optimally. In particular, prior theoretical work has only solved the CAM problem in the case where all jobs follow the same speedup function, with very little progress on multiple speedup functions.

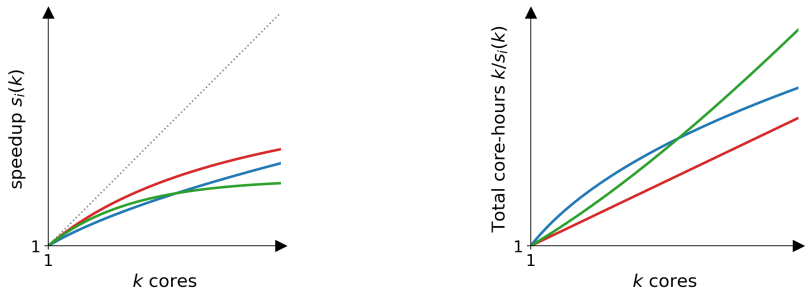
This paper takes a new approach to CAM, looking at the problem in the mean field asymptotic regime. In this regime, we are able to optimally solve a much more general instance of CAM. We allow for any number of job classes, each with an arbitrary concave speedup function and weight. Furthermore, unlike prior work, we allow for general independent inter-arrival times and phase-type job sizes.

We derive two distinct mean field optimal policies, FW-CAM and GLAM. FW-CAM demonstrates a new intuition: in the mean field regime, individual job sizes are not relevant in finding an optimal policy. GLAM (Gittins-Like Allocation for Malleable jobs) is asymptotically optimal and *also* serves as a good heuristic outside of the asymptotic regime by adapting dynamically to the current number of jobs in the system. Notably, none of the policies previously proposed in the literature are mean field optimal when jobs follow different speedup functions.

1. Introduction

Modern data centers and cloud computing clusters are increasingly tasked with hosting workloads composed of *malleable* jobs, such as machine learning training tasks [26, 41], large-scale database queries [33, 45], and data center computing tasks [43, 13]. Unlike traditional single-server jobs, malleable jobs can utilize multiple processor cores to accelerate their execution. Furthermore, unlike multi-server jobs that require a specific number of cores to run, malleable jobs do not have hard constraints on the degree of parallelism; a malleable job can execute on *any* number of cores, and can change its degree of parallelism as it runs. Given a stream of malleable jobs that arrive to a system over time, a system operator must allocate a fixed pool of n cores across the arriving jobs to optimize system performance (e.g., the mean response time, where a job's response time is the time from when it arrives until it completes). Designing an efficient *Core Allocation Policy* for this setting remains a significant open challenge.

Motivated by these applications, this paper studies the problem of *Core Allocation to Malleable jobs* (which we call CAM). In this problem, we are given n cores and a stream of malleable jobs. The goal is to, at all times, determine how many cores to assign to each job – the number of cores assigned to a job can change over time. The objective is to minimize the (weighted) mean response time (see Section 3).



(a) Speedup functions $s_i(k)$.

(b) Total core-hours $k/s_i(k)$.

Figure 1: Examples of speedup functions and the corresponding total core-hours (assuming job size is 1). Speedup functions are typically sublinear and concave; $k/s(k)$ is increasing and at least 1, but it is not restricted to being convex or concave.

1.1. Speedup functions and their effect on running time

A defining characteristic of this problem is the diminishing marginal returns associated with parallel execution. A job's *speedup function*, $s(k)$, describes how much faster a job runs on k cores compared to a single core. Job speedup functions are typically concave and sublinear (see Figure 1a for some examples). Hence running a job on 4 cores may only increase its speed by a factor of 2.

The CAM problem is complex because there are multiple classes of jobs, where every job class is associated with a different speedup function. Specifically, $s_i(k)$ denotes the speedup function of a class-i job (see Section 3 for the formal definition). Additionally, every class of job can have a different *size* distribution. The *size* of a job is the inherent work associated with the job, which we measure as the job's running time on a single core. The *running time* of a job of size x and class i on k cores is given by $\frac{x}{s_i(k)}$, as explained in Section 3.

In the CAM problem, as more cores are allocated to a single job, the overall system efficiency (defined as the total inherent work completed per unit of core time) decreases. This efficiency loss is captured by the distinction between *system load* and *effective load*, defined as follows.

For an m-class GI/GI/n system (n cores), the *system load*, ρ, is defined as

$$\text{system load} = \rho = \frac{\sum_{i=1}^{m} \lambda_i \mathbf{E}[X_i]}{n} < 1,$$

where λ_i is the arrival rate of class-i jobs, X_i is a random variable denoting the job size of a class-i job, and n is the number of cores. While the system load can be defined exactly the same way for the CAM problem, the system load alone no longer captures the queueing behavior of the system. This is because parallelization increases the "effective work" of a job due to sublinear speedup (see Figure 1b for an example). For instance, if a job runs on 4 cores but achieves only a 2× speedup, it consumes twice the core-seconds compared to the case when it is run on 1 core, or, equivalently, the job's "effective work" is double its inherent work. Consequently, the *effective load* on the system is strictly greater than the inherent system load, and the system may become unstable even when the system load is smaller than 1. Thus, the queueing behavior of a system is truly determined by the *effective load*, which depends on both the system load and the core allocation policy.

1.2. A Brief History of the CAM Problem

A variety of work from the systems community considers scheduling malleable jobs [37, 41, 26, 43, 45]. However, all of these systems rely on simple heuristic policies with no theoretical guarantees.

The CAM problem has been studied by the performance modeling community over the past decade. Much of this work assumes exponential job sizes and Poisson arrivals [5, 6, 8].

There are several papers devoted to the case where all jobs follow a *single* speedup function [5, 6, 9]. Even the single speedup function problem is difficult since one needs to balance two competing goals: (1)

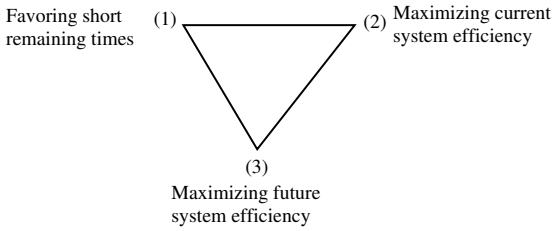


Figure 2: Three competing priorities when dealing with multiple speedup functions.

giving more cores to jobs with short remaining times, and (2) maximizing system efficiency. The optimal balance is achieved in the heSRPT paper [9], which assumes all jobs are present at the start. The problem is still open when jobs arrive over time.

Once one moves to multiple speedup functions, the problem becomes even harder. Berg et al. [5] suggest that, in general, one must trade off three goals, shown in Figure 2: (1) favoring jobs with short remaining times, (2) maximizing current system efficiency, and (3) maximizing future system efficiency (specifically "saving up" some highly parallelizable jobs for times when there are only a few jobs in the system). Prior works have looked at special cases that reduce to only two out of the three tradeoffs. For example, [7, 10] consider the case of two speedup functions, where the shorter jobs are also the less parallelizable jobs, aligning corners (1) and (3). As another example, [8] assumes all speedup functions follow a *rooftop* shape: class-i jobs are perfectly parallelizable up to k_i cores, but get no further benefit from $> k_i$ cores. There, it is easy to satisfy corner (2), by avoiding the flat parts of the speedup functions, but trading off between (1) and (3) still requires looking at the case of $\rho \to 1$.

In summary, the CAM problem with multiple speedup functions is still largely wide open.

1.3. Asymptotic Regimes

In Section 1.2, we saw that an optimal solution to the CAM problem is elusive, since it must trade off several competing goals. In fact, we note here that the CAM problem is strictly more challenging than the classical M/G/n scheduling problem (where jobs are not parallelizable and each job occupies a single server). The optimal scheduling policy for the M/G/n remains open except under asymptotic regimes (see Section 2 for more discussion). Hence, in order to build intuition for a good allocation policy for CAM, we too turn to asymptotic analysis.

We limit our attention to the wide class of regimes where both the arrival rate and number of cores go to infinity while the job sizes stay fixed. In traditional M/G/n systems, the regimes are differentiated by how the arrival rate scales compared to the number of cores. For example, in the *mean field* regime [12, 18, 17, 16], the arrival rate scales proportionately to the number of cores; hence, the system load ρ is held fixed. This is a good representation for real-world large-scale systems that operate under reasonable (not too high) loads. By contrast, in non-mean-field regimes, the arrival rate scales faster than the number of cores, causing the system load to approach 1 ([25, 24]).

In the CAM problem, running jobs on more than 1 core causes the effective load to be higher than the system load. Hence for the CAM problem, regimes where the system load ρ approaches 1 are not interesting because they do not allow jobs to be parallelized. Specifically, given jobs that follow strictly sublinear speedup functions, if any constant fraction of jobs run on more than 1 core, the effective load is a constant factor larger than the system load (which is already approaching 1), resulting in an effective load greater than 1 and system instability. We formalize this intuition in Proposition 1. This observation compels us to focus on the *mean field* asymptotic regime (see Section 3), where the system load remains constant while the number of cores scales.

1.4. Our Approaches and Impacts

We derive two distinct mean-field optimal policies, **FW-CAM** and **GLAM**, using fundamentally different methodologies.

FW-CAM (Definition 1): In prior work, the tradeoffs shown in Figure 2 are difficult to balance because the importance of each conflicting goal varies with the system state. Our key observation is that in the mean field limit, the *average* job arrival rates, job sizes, and speedup functions for each class are the only quantities that matter. This allows us to derive a mean field optimal policy, FW-CAM, as a function of these averages.

While conceptually simple, FW-CAM reveals a critical insight: a mean field optimal policy should treat all jobs in the same class (i.e., having the same speedup function) equally, allocating cores independently of the job's remaining size. That is, every job class is associated with a *fixed width* (FW). This contradicts the classical scheduling intuition of "prioritizing short jobs" (e.g., [9, 8]). In fact, any policy that does prioritize short jobs is **not** mean field optimal (see Proposition 2).

GLAM (Definition 5): While FW-CAM successfully exploits the averaging effect of the mean-field regime, its allocations are static over time. As a result, we find that FW-CAM's performance degrades outside the asymptotic limit (if the number of cores, n, is small), raising questions about the practicality of using FW-CAM in the real world. This motivates us to introduce a second policy GLAM (Gittins-Like Allocation policy for Malleable jobs) that is both mean-field optimal *and* adapts to the state of the system over time.

GLAM uses a heuristic based on the Gittins index [20] to form a dynamic, mean-field optimal policy. While the Gittins index has classically been used for scheduling problems with non-parallelizable jobs, we use the approach to handle malleable jobs. Roughly speaking, GLAM computes every job's marginal improvement in response time from running on an additional core. Running on an additional core also increases the number of core-hours the job will use (see Figure 1b), so GLAM also computes the marginal increase in resource usage from an additional core. GLAM favors jobs with a higher ratio of marginal response time improvement to marginal resource usage increase.

Notably, GLAM uses a metric that does not cleanly align with any of the tradeoffs described in Figure 2. In particular, it is not maximizing the classical definition of system efficiency. In fact, we can show that the GREEDY policy [5] that maximizes system efficiency is not mean field optimal (see Section 7 and Appendix I). Furthermore, GLAM (like FW-CAM) does not favor short jobs in mean field.

1.5. Our Contributions

The contributions of this paper are as follows:

- We derive two mean field optimal policies, FW-CAM and GLAM, for the CAM problem under general settings. In particular, we allow for generally-distributed independent inter-arrival times, phase-type job sizes, multiple job classes with distinct speedup functions, and any weighted mean response time metric.
- (Section 5) Our first policy, FW-CAM, leverages the concept of effective load to derive a mean field optimal policy, where all jobs in the same class get the same allocation.
- (Section 6) Our second policy, GLAM, serves as a bridge between theory and practice. It is not only mean field optimal but also demonstrates superior empirical performance, converging rapidly to the optimal limit. It serves as a strong heuristic for general CAM settings.
- Prior work on the CAM problem has been largely limited to only *one* speedup function or a narrow class of speedup functions (see Section 2). Even for the setting of a single speedup function, our analysis provides novel insights challenging the conventional wisdom of prioritizing short jobs.

2. Prior Work

In Section 1.2, we already reviewed the prior work on the CAM problem. In this section, we discuss prior work on scheduling parallelizable jobs under different models of parallelism that are complementary to the CAM problem.

2.1. Scheduling problems relevant to, but different from CAM

Even under traditional multi-server scheduling, where jobs are not parallelizable and each job occupies only a single server, the optimal policy is still widely open. For M/G/n systems, the only known optimality result applies in the conventional heavy traffic regime [21]. However, it has recently been shown that SRPT-n is *not* optimal under non-limiting load [23], and this line of work is still active.

Another active line of work on multi-server scheduling involves *multiserver jobs* (e.g. [22, 11]). These jobs are parallelizable, but not malleable. Specifically, in [22, 11] each job requires a fixed given number of cores specific to that job. It is hard to translate intuitions learned from the multiserver job setting to the CAM setting where jobs are malleable and can run on *any* number of cores.

Additionally, there has been significant work on *fork-join parallelism* [28, 4], where a single job is composed of a fixed number of tasks that queue and run independently. A single job is completed when all of its tasks are completed. Analysis of fork-join queueing systems is notoriously difficult, and known results are generally limited to approximations [38] or asymptotic analysis [46].

Each of the above models is important for some range of systems. However, none of these models capture the many systems where jobs are malleable, and the scheduler chooses the level of parallelism for each job. We do note that many of the considerations and tradeoffs that arise in the CAM problem can be seen in some form for different parallel scheduling models. However, none of the results from these other models directly generalize to provide results in the CAM model.

2.2. The Gittins Index

In the latter half of this paper, we use techniques related to the Gittins index [20] to derive a mean field optimal policy for the CAM problem. The Gittins index has been used for optimal scheduling in an M/G/1 queue [1, 42, 2] where job sizes are unknown, but the job size distribution is known. The Gittins index was further used in an M/G/n setting, where it was shown to minimize mean response time in the heavy-traffic limit ($\rho \to 1$)[21]. Importantly, the Gittins index is not optimal outside of heavy-traffic for the M/G/n. Our work is the first to use a Gittins index approach for scheduling jobs governed by speedup functions. Our model considers jobs of known size, while classical scheduling applications of the Gittins index focus on jobs with unknown sizes. Hence, our work should be viewed as a generalization of SRPT-like policies rather than a generalization of prior Gittins policies for the M/G/1 and M/G/n.

3. Core Allocation to Malleable jobs: Our Problem Setting

We consider an m-class GI/GI/n system with n homogeneous cores.

3.1. Job Classes

There are m classes of jobs. Each class has its own arrival process, job size distribution, speedup function and weight, all to be defined below.

Arrival process. The arrival process of class i is defined by i.i.d. inter-arrival times drawn from distribution A_i. We assume that A_i has mean $\frac{1}{\lambda_i}$. Define $\lambda := \sum_{i=1}^{m} \lambda_i$ to be the total arrival rate of jobs and let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_m)$ be the vector of arrival rates for each job class. Let $\mathbb{A}_i(t)$ be the number of class-i jobs that have arrived by time t and let $\mathbb{A}(t) = \sum_{i=1}^{m} \mathbb{A}_i(t)$ denote the total number of arrivals by time t.

Job size distribution. The *size* of a job is defined as the time needed to complete a job on a single core. We can think of this as the *inherent work* associated with the job. The sizes of class i jobs are i.i.d. random variables, X_i, with $\mathbf{E}[X_i] = 1/\mu_i$. Throughout, we assume job sizes are known, but surprisingly, the exact size of each job turns out to be useless in the mean field regime. While our results on FW-CAM hold for general, independent job sizes, our results on GLAM require phase-type job sizes.

Speedup function. A job's running time depends on the number of cores the job runs on and on its *speedup function*. We define $s_i : \mathbb{R}^+ \to \mathbb{R}^+$ to be the *speedup function* of class i jobs. If a class i job with size x runs on k cores, then its running time becomes

$$\text{Running time on } k \text{ cores} = \frac{x}{s_i(k)}.$$

Note that we assume that jobs can be assigned fractional cores, but a job cannot be assigned between 0 and 1 core. We make the following mild assumptions on the speedup functions: For any job class i, the speedup function $s_i(\cdot)$ should fulfill the following properties:

- $s_i(1) = 1$.
- Sub-linearity: $s_i(k) \leq k$ for $k \geq 1$.
- Monotonicity: for any $1 \leq k_1 < k_2$, we have that $s_i(k_1) \leq s_i(k_2)$.
- Concavity: for any $1 \leq k_1 < k_2$, we have that $s_i(k_1)/k_1 \geq s_i(k_2)/k_2$.
- Smoothness: $s_i'(k)$ exists for all $k > 1$.

Although real-world speedup functions may not be perfectly concave, in practice, this is dealt with by using the monotonic, concave hull of an empirically measured speedup function [19].

Our second mean-field optimal policy, GLAM, needs the following additional assumptions:

- Strict sub-linearity: $\lim_{\epsilon \to 0^+} \frac{s_i(1+\epsilon) - s_i(1)}{\epsilon} < 1$ and $s_i(k) = o(k)$ for $k \to \infty$.
- Strict concavity: $s_i''(k)$ exists and $s_i''(k) < 0$ for any $k > 1$.

Stability Conditions. We assume some additional conditions on the job size distribution and interarrival distribution that are necessary for system stability. In addition to the standard assumption that system load is less than 1 (see Section 3.3), additional distributional assumptions are required to guarantee stability. In particular, the stability conditions for the CAM system mirror those of a GI/GI/n queue. A full treatment of these stability conditions is given in [15], and is beyond the scope of this paper. In this paper, we assume $\mathbf{E}\left[A_i^2\right] < \infty$ and $\mathbf{E}\left[X_i^3\right] < \infty$ for all i, along with all other conditions on A_i and X_i that are needed for stability.

3.2. Performance Metrics

Weighted mean response time. Our goal is to minimize the weighted mean response time across jobs, where class i is assigned weight c_i. Formally, let T_i be the stationary response time of class-i jobs with mean $\mathbf{E}[T_i]$. The *weighted mean response time* of the system is given by

$$\mathbf{E}\left[T_{\text{Weighted}}\right] := \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} c_i \mathbf{E}\left[T_i\right]. \tag{1}$$

No Preemption Overhead. We assume that there is no overhead when preempting or changing a job's allocation. This simplification is also assumed in most prior theory works, e.g., [7, 8, 10, 19, 9].

Optimizing weighted mean response time. Let $k_{ij}(t)$ be the number of cores allocated to the jth class-i job at time t. Let T_{ij} be the response time of the jth class-i job. Observe that T_{ij} is determined by the choices of $k_{ij}(t)$ at every time t. We can then write the following optimization problem, which describes the problem of choosing job allocations to minimize the weighted mean response time across jobs in the CAM problem.

$$\begin{aligned}
\underset{k_{ij}(t)}{\text{minimize}} \quad & \lim_{\tau\to\infty} \frac{1}{\mathbb{A}(\tau)} \sum_{i=1}^{m} \sum_{j=1}^{\mathbb{A}_i(\tau)} T_{ij} c_i \qquad \text{(weighted mean response time)} \\
\text{subject to} \quad & \sum_{i=1}^{m} \sum_{j=1}^{\mathbb{A}_i(\tau)} k_{ij}(t) \le n, \quad \forall t \ge 0 \qquad \text{(usage constraint at time } t\text{)} \\
& k_{ij}(t) \in \{0\} \cup [1, \infty) \quad \forall t \ge 0.
\end{aligned} \tag{2}$$

We will refer back to this optimization problem frequently as a helpful starting point for deriving new scheduling policies.

3.3. The Mean Field Regime

The main results of this paper consider the *mean field scaling regime*. In the mean field regime, the number of classes, job size distributions and speedup functions are held fixed, and the arrival rate is scaled proportionally with the number of cores. Formally, a sequence of systems indexed by d are considered as $d \to \infty$. Let $\lambda_i^{(d)}$ be the arrival rate of type-i jobs in the d^{th} system, and let $n^{(d)}$ be the number of cores in the d^{th} system. Then in the mean field regime, $\lambda_i^{(d)} = d \cdot \lambda_i$, $\boldsymbol{\lambda}^{(d)} = d \cdot \boldsymbol{\lambda}$, and $n^{(d)} = d \cdot n$. More precisely, inter-arrival times in the d^{th} system are sampled from $A_i^{(d)} := \frac{A_i}{d}$. In this way, the mean inter-arrival time scales, but the coefficient of variation of the inter-arrival time distribution stays constant. Importantly, in the mean field regime, the *system load* stays constant as d is increased. Mathematically, the system load in the d^{th} system is defined as

$$\rho := \sum_{i=1}^{m} \rho_i \quad \text{where} \quad \rho_i := \frac{1}{n^{(d)}} \lambda_i^{(d)} \cdot \mathbf{E}\left[X_i\right].$$

Neither ρ nor ρ_i depend on d. We assume throughout the paper that $\rho < 1$, which is necessary for system stability.

We refer to the CAM problem in the d^{th} system as $CAM^{(d)}$. We denote the weighted mean response time of a policy π in the CAM$^{(d)}$ system by $\mathbf{E}\left[T_{\text{Weighted}}\right]^{\pi}_{CAM^{(d)}}$. The policy π is considered *mean field optimal* for the CAM problem if, for any other policy π', we have

$$\lim_{d\to\infty} \frac{\mathbf{E}\left[T_{\text{Weighted}}\right]^{\pi}_{CAM^{(d)}}}{\mathbf{E}\left[T_{\text{Weighted}}\right]^{\pi'}_{CAM^{(d)}}} \le 1.$$

4. Preliminaries: A relaxed version of the CAM problem

In this section, we introduce a relaxed version of the CAM problem, which provides a lower bound on the optimal weighted mean response time, $\mathbf{E}\left[T_{\text{Weighted}}\right]^*$, for the CAM problem. In the Relaxed problem, we drop the hard constraint that at every moment of time only n cores are available. Instead, we allow the total number of cores to fluctuate at our control, so long as the time-average number of cores stays within a budget of n. Similar relaxations are commonly used in queueing problems to prove asymptotic optimality [44, 47, 48].

Specifically, consider the jth job of class i that arrives at time t. Let

$$\overline{k}_{ij} = \frac{\int_t^{t+T_{ij}} k_{ij}(s) ds}{T_{ij}}$$

be the average number of cores used to process the jth job of class i. Looking at the optimization problem in (2), we relax the constraint on the total number of cores used at time t to

$$\frac{1}{\tau} \sum_{i=1}^{m} \sum_{j=1}^{\mathbb{A}_i(\tau)} \overline{k}_{ij} T_{ij} \le n. \tag{3}$$

It is straightforward to see that this new constraint is weaker than the original constraint in (2), since the original constraint clearly implies that the time average core usage is below n. Hence, the weighted mean response time for the Relaxed problem is a lower bound on the optimal weighted mean response time in the CAM problem, since the constraint of a fixed number of cores is relaxed. Formally, we denote the optimal weighted mean response time for the CAM problem and the Relaxed problem by $\mathbf{E}\left[T_{\text{Weighted}}\right]^*_{CAM}$ and $\mathbf{E}\left[T_{\text{Weighted}}\right]^*_{Relaxed}$, respectively. Then we have that

$$\mathbf{E}\left[T_{\text{Weighted}}\right]^*_{CAM} \geq \mathbf{E}\left[T_{\text{Weighted}}\right]^*_{Relaxed}. \tag{4}$$

As it turns out, the Relaxed problem is of independent theoretical interest, and was previously considered in [35]. However, [35] does not consider mean-field scaling nor does it address applications to the original CAM problem. Instead, [35] solves the Relaxed problem by deriving two results. First, they show that the optimal policy for the Relaxed problem should never queue jobs; intuitively, queueing any job makes its response time worse without reducing the time-average number of cores it will eventually use. Second, [35] shows that the optimal Relaxed policy should not change a job's core allocation over the course of the job's lifetime. This is a consequence of the concavity of the job's speedup function — a job whose allocation changes over time will run faster by receiving a constant allocation equal to its time-average core allocation. A similar argument can be used to show that two different jobs of the same class should receive equal allocations. Taken together, these properties imply that one can find the optimal policy for the Relaxed problem by computing the optimal number of cores, k_i, to allocate to every class-i job. This k_i will replace the $\overline{k}_{ij}$ in our Relaxed optimization problem, (3). The optimal Relaxed policy then allocates the appropriate number of cores to each arriving job with no queueing.

Given this structure of the optimal Relaxed policy, it is straightforward to obtain simplified expressions for both the weighted mean response time from (2) and the relaxed core usage constraint in (3). This result is summarized in Theorem 1, as follows.

Theorem 1 (Theorem 1 in [35]). *The optimal policy for the Relaxed problem is: Whenever a class-i job arrives in the system, k_i cores are immediately allocated to it until it completes, where k_i is the solution to the following convex optimization problem*

$$\begin{aligned}
\underset{k_i}{\textit{minimize}} \quad & \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} \mathbf{E}\left[T_i\right] c_i = \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} \frac{\mathbf{E}\left[X_i\right] c_i}{s_i(k_i)} && \textit{(minimize weighted mean response time)} \\
\textit{subject to} \quad & \sum_{i=1}^{m} \lambda_i \mathbf{E}\left[T_i\right] k_i = \sum_{i=1}^{m} \frac{\lambda_i \mathbf{E}\left[X_i\right] k_i}{s_i(k_i)} \leq n, && \textit{(time-average budget constraint)} \\
& k_i \geq 1.
\end{aligned} \tag{5}$$

The convexity of (5) *is shown in Appendix A.*

We refer to the optimal policy for the Relaxed problem as the *Relaxed policy*. We note that the Relaxed policy is an example of a *fixed-width policy* (FW for short) because every job of class i gets the same number of cores, k_i.

We note that, even in the Relaxed model, the above results are counterintuitive. Because the optimal policy does not queue jobs, the typical benefits of prioritizing short-running jobs ahead of long-running jobs disappear. Specifically, given two jobs with the same speedup function but vastly different job sizes, the optimal policy will allocate the same number of cores to each job, contradicting the tradeoffs in Figure 2.

We now apply Theorem 1 in the mean field regime where $\lambda_i^{(d)} = d \cdot \lambda_i$ and $n^{(d)} = d \cdot n$. Importantly, the optimization problem in (5) when written for the dth system, is independent of d. Hence, (5) is also a lower bound on the weighted mean response time of the mean-field optimal policy for CAM.

5. The first Mean Field optimal policy: FW-CAM

In this section, we present the first mean field optimal policy for the CAM problem, FW-CAM. In Section 5.1 we first develop the intuition behind our policy. Next in Section 5.2 we present our FW-CAM policy. Finally in Section 5.3 we present the proof of its mean field optimality.

5.1. Intuition behind FW-CAM

The goal of this section is to use the Relaxed policy (see Section 4) to derive an asymptotically optimal policy for the CAM problem, FW-CAM. There are two issues with doing this.

First, the Relaxed policy is allowed to use more than n cores during some periods of time. In the CAM system, jobs would be forced to queue during these periods.

Second, if the Relaxed policy is directly used in the CAM system, allocating k_i cores to every class-i job, the resulting CAM system will be unstable. To see this, consider the *effective load* of the Relaxed policy in the CAM system. When each class-i job runs on k_i cores, it consumes $\mathbf{E}[T_i]\, k_i = \frac{\mathbf{E}[X_i]k_i}{s_i(k_i)}$ amount of core time (i.e., CPU-seconds) in expectation. Hence, the total effective load of the Relaxed policy in the CAM system is

$$\text{Total Effective Load} = r = \sum_{i=1}^{m} r_i \qquad \text{where} \qquad r_i = \frac{\lambda_i}{n} \cdot \frac{\mathbf{E}[X_i]\, k_i}{s_i(k_i)}$$

Here, r_i represents the effective load contributed by class i jobs.

Because the budget constraint in (5) will be tight when deriving the optimal Relaxed policy, the effective load of the optimal Relaxed policy will be 1 in the CAM system, leading to system instability.

The above issues motivate the rest of the section. Our goal is to reduce the effective load of the Relaxed policy in the CAM system such that queueing disappears. We then argue that, in the mean field scaling regime, we can make this change without significantly reducing the number of cores allocated to each job, and hence without impacting performance. The resulting policy will be mean-field optimal because it will perform similarly to the Relaxed policy in the Relaxed system, whose weighted mean response time is a lower bound on the weighted mean response time of the CAM system.

5.2. Defining FW-CAM

We begin by formally defining FW-CAM, which will adapt the Relaxed policy to the CAM system. Roughly speaking, under FW-CAM, we set aside a small number of cores (on the order of $n^{0.75}$). With the remaining $n^{(d)} - \left(n^{(d)}\right)^{0.75}$ cores, we solve the optimization problem (5) to find the optimal Relaxed policy that allocates the $n^{(d)} - \left(n^{(d)}\right)^{.75}$ cores on average. We then apply the resulting Relaxed policy in the original CAM system. The resulting policy uses only $n^{(d)} - \left(n^{(d)}\right)^{0.75}$ out of the n total cores on average. Hence, the effective load of FW-CAM is less than 1 and the system is stable. As d increases, the effective load of FW-CAM in the d^{th} system scales as roughly $1 - d^{-0.25}$. This scaling regime lies within the Sub-Halfin-Whitt regime[1]. We will show that the mean queueing time in the CAM system goes to 0 in this scaling regime.

We start by defining FW-CAM for an arbitrary CAM system with a vector of arrival rates $\boldsymbol{\lambda}$ and n cores.

Definition 1 (FW-CAM). Let $k_i^*(\boldsymbol{\lambda}, n)$ be the optimal value of k_i from the optimization problem (5), given a vector of arrival rates, $\boldsymbol{\lambda}$, and a time-average budget constraint of n cores.[2] Pick $0.75 < \beta < 1$ and let $k_i := k_i^*\left(\boldsymbol{\lambda}, n - n^\beta\right)$ be the number of cores allocated by FW-CAM to each class-i job. We partition the n cores into m groups, where class i's group consists of n_i cores. Here n_i is defined to be proportional to the effective load contributed by class i. That is,

$$n_i := n \cdot \frac{r_i}{\sum_{j=1}^{m} r_j}.$$

Whenever a class i job arrives, if the total number of cores used by existing class i jobs is no more than $n_i - k_i$, the new job is allocated k_i cores until it completes; otherwise, the new job queues. Whenever a class

[1] The Sub-Halfin-Whitt regime refers to systems where load scales as $1 - \omega(d^{-0.5})$. We enforce a stronger scaling regime $(1 - \omega(d^{-0.25}))$ in order to apply the results in [34].

[2] We assume the optimization (5) admits an optimal solution. The corner case where the infimum is not attained at any finite k_i is not realistic but, for theoretical completeness, is handled in Appendix B.

i job completes, if there are queueing class i jobs, the released k_i cores are assigned to the class-i job that has waited the longest; otherwise, if there are no queueing class i jobs, the released cores idle even if there are queueing jobs from other classes.

Note that n_i is chosen so that class i jobs experience a system with subcritical effective load. Statically reserving cores in this way is not necessary for implementation, but greatly simplifies our analysis.

5.3. The mean field optimality of FW-CAM

We now prove the mean field optimality of FW-CAM. Our proof has two major components. First, we show in Lemma 1 that, although the allocations of FW-CAM are slightly lower than the allocations of Relaxed (because FW-CAM has reserved some cores), the allocations used by FW-CAM converge to the allocations used by the Relaxed policy in the mean field limit. Hence, the *weighted service time* component of weighted mean response time under FW-CAM converges to that under the Relaxed policy. Second, we show in the remainder of the section that the mean *queueing time* under FW-CAM converges to 0. Taken together, these results show that the mean response time under FW-CAM converges to the mean response time of the Relaxed policy.

We begin with the following lemma that compares the allocations of FW-CAM to the Relaxed policy in the mean field limit when both systems have $n^{(d)}$ cores and a vector of arrival rates scaled up by d, which we denote as $\boldsymbol{\lambda}^{(d)}$. This lemma will be used in Lemma 4 and Theorem 2.

Lemma 1. *Let* $k_i^{(d)} := k_i^*\left(\boldsymbol{\lambda}^{(d)}, n^{(d)} - \left(n^{(d)}\right)^\beta\right)$. *For any class* i, $\lim_{d\to\infty} k_i^{(d)} = k_i^*(\boldsymbol{\lambda}, n)$. *That is, the allocations under FW-CAM converge to the allocations of the Relaxed policy in the mean field limit.*

Proof (Proof Sketch). Note that the optimization problem (5) does not change when both the $\boldsymbol{\lambda}$ and n are scaled by the same constant, d. This gives

$$k_i^*\left(\boldsymbol{\lambda}^{(d)}, n^{(d)} - \left(n^{(d)}\right)^\beta\right) = k_i^*\left(\frac{\boldsymbol{\lambda}^{(d)}}{d}, \frac{n^{(d)} - \left(n^{(d)}\right)^\beta}{d}\right) = k_i^*\left(\boldsymbol{\lambda}, n - n^\beta d^{\beta-1}\right).$$

Because $\beta < 1$ we have

$$\lim_{d\to\infty} n - n^\beta d^{\beta-1} = n.$$

The lemma then follows from the continuity and convexity of (5). See Appendix E.

To show that the mean queueing time under FW-CAM goes to 0 in the mean field limit, it helps to first see how several of the critical quantities in the system will scale with d under the CAM policy. In particular, we first define the effective load contributed by class i jobs in the dth system

$$\text{Effective load contributed by class } i = r_i^{(d)} = \frac{\text{expected cores used by class } i}{n^{(d)}} = \frac{\lambda_i^{(d)}}{n^{(d)}} \cdot \frac{\mathbf{E}[X_i]\, k_i^{(d)}}{s_i(k_i^{(d)})}. \tag{6}$$

We can then examine the asymptotic behavior of the total effective load as follows:

$$\text{Expected total no. cores used } = \sum_{i=1}^m \frac{\lambda_i^{(d)} \mathbf{E}[X_i]\, k_i^{(d)}}{s_i(k_i^{(d)})} = n^{(d)} - \left(n^{(d)}\right)^\beta.$$

$$\text{Total effective load } = \sum_{i=1}^m r_i^{(d)} = \frac{\text{total expected cores used}}{n^{(d)}} = 1 - \left(n^{(d)}\right)^{\beta-1}. \tag{7}$$

By definition,

$$n_i^{(d)} = n^{(d)} \cdot \frac{r_i^{(d)}}{\sum_{j=1}^m r_j^{(d)}}.$$

We now analyze the subsystem composed of the $n_i^{(d)}$ cores used to process class-i jobs. Note that the cores are split into classes proportionately to the effective load, hence the effective load on this subsystem is the same as the total effective load. Mathematically, we have

$$\text{Effective load on class-i subsystem } = \frac{\lambda_i^{(d)}\left(\frac{\mathbf{E}[X_i]k_i^{(d)}}{s_i(k_i^{(d)})}\right)}{n_i^{(d)}} = \frac{r_i^{(d)}n^{(d)}}{n_i^{(d)}} = \sum_{i=1}^{m} r_i^{(d)} = 1 - \left(n^{(d)}\right)^{\beta-1}. \quad (8)$$

Hence, each job class experiences a subsystem that follows a Sub-Halfin-Whitt scaling regime that matches the overall system.

We now show that the mean queueing time under FW-CAM goes to 0 in the mean field limit. To do this, we start with Lemma 2, which shows that the system under FW-CAM can be decoupled into m independent FCFS (first-come-first-serve) GI/GI/n_i' systems where $n_i' = \lfloor \frac{n_i}{k_i} \rfloor$.

Lemma 2. *Let $n_i'^{(d)} = \left\lfloor \frac{n_i^{(d)}}{k_i^{(d)}} \right\rfloor$. Then the mean response time of class-i jobs under FW-CAM in the $CAM^{(d)}$ system is the same as the mean response time in a single-class GI/GI/$n_i'^{(d)}$ system with average arrival rate $\lambda_i^{(d)}$, job sizes distributed as $\frac{X_i}{s_i(k_i^{(d)})}$ and system load, $r_i'^{(d)}$, given by*

$$r_i'^{(d)} := \frac{\lambda_i^{(d)}\left(\mathbf{E}\left[X_i\right]/s_i\left(k_i^{(d)}\right)\right)}{n_i'^{(d)}}. \quad (9)$$

Proof. The proof is straightforward. Under FW-CAM, each job class sees an independent system and each class-i job occupies $k_i^{(d)}$ cores. Thus, the running time of each class-i job follows the distribution $\frac{X_i}{s_i(k_i^{(d)})}$. The $n_i^{(d)}$ cores reserved for class-i jobs are divided into chunks of size $k_i^{(d)}$. Here, $n_i'^{(d)}$ is the number of complete chunks of size $k_i^{(d)}$, hence the floor. This creates a natural coupling between the system under FW-CAM and the GI/GI/n_i' system described above. The load on the resulting system reflects the resources consumed by class-i jobs (i.e., chunk-seconds used) divided by the number of chunks used.

We now apply Lemma 3, adapted from [34], that bounds the mean number of jobs in queue in a GI/GI/n system. Our statement below is simplified from the original statement in [34] to aid in readability. The full Lemma is provided in Appendix F.

Lemma 3 (Adapted from Corollary 3 in [34]). *For a GI/GI/$n^{(d)}$ queue with arrival rate $\lambda^{(d)}$ and job size distribution S, let the system load, $\rho^{(d)}$, be defined as $\rho^{(d)} := \frac{\lambda^{(d)}\mathbf{E}[S]}{n^{(d)}} < 1$ and let $\mathbf{E}\left[N_Q^{(d)}\right]$ be the mean number of jobs in queue. If $\rho^{(d)} = 1 - \omega\left(\left(n^{(d)}\right)^{-.25}\right)$, then*

$$\lim_{d\to\infty} \mathbf{E}\left[N_Q^{(d)}\right] = 0.$$

That is, if the system load of the GI/GI/$n^{(d)}$ scales in a strongly Sub-Halfin-Whitt regime, the mean number of jobs in the queue goes to 0.

We now use this result to finish our proof of the mean field optimality of FW-CAM.

Lemma 4 (No queueing for each class). *For any class i, the expected queueing time of class-i jobs under FW-CAM goes to 0 when $d \to \infty$.*

PROOF. By Lemma 2, the mean queueing time of class-i jobs is equal to the mean queueing time in the GI/GI/$n_i'^{(d)}$ system described in Lemma 2. Thus, we only need to show that the mean queueing time in the GI/GI/$n_i'^{(d)}$ system goes to 0 when $d \to \infty$. It is straightforward to see that $r_i'^{(d)}$ from (9) has the same asymptotic behavior as the effective load on the class-i subsystem as described in (8). Hence, by Lemma 3, the expected queueing time in the GI/GI/$n_i'^{(d)}$ system goes to 0 as $d \to \infty$ (using Little's Law).

Finally, we can prove our main theorem, which shows the mean field optimality of FW-CAM.

Theorem 2. *Our policy FW-CAM is mean field optimal.*

PROOF. By Lemma 1, as $d \to \infty$, the number of cores allocated to a class-i job converges to $k_i^*(\boldsymbol{\lambda}, n)$. Hence the expected service time of class-i jobs converges to $\frac{\mathbf{E}[X_i]}{s_i(k_i^*)}$. Moreover, by Lemma 4, the expected queueing time of class-i jobs converges to 0. Thus, the expected response time of class-i jobs converges to $\frac{\mathbf{E}[X_i]}{s_i(k_i^*)}$, and the weighted mean response time converges to

$$\lim_{d\to\infty} \mathbf{E}\left[T_{\text{Weighted}}\right]_{CAM^{(d)}}^{FW-CAM} = \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} c_i \frac{\mathbf{E}\left[X_i\right]}{s_i(k_i^*)},$$

which is the weighted mean response time lower bound for CAM established in (4).

6. A "Better" Mean-Field Optimal policy: GLAM

In Section 5, we prove the mean field optimality of FW-CAM. However, outside the mean field regime, FW-CAM may make frequent mistakes. For example, if the number of jobs in the system is very small, FW-CAM may leave many cores idle. On the other hand, if there are too many jobs in the system, FW-CAM allocates many cores to a subset of jobs, while forcing other jobs to queue. Intuitively, outside the mean field regime, a better policy should take the *current number of jobs in the system* into consideration. This raises the question of whether we can develop a policy that is more dynamic than FW-CAM while remaining mean field optimal.

For example, prior work [5] proposed the GREEDY policy[3] that accounts for the number of jobs in the system by trying to maximize the sum of the speedups of all jobs in the system at every moment in time. Specifically, GREEDY maximizes the current *system efficiency* by repeatedly allocating cores to the jobs that receive the highest marginal improvement in speedup. Using this heuristic, GREEDY tends to spread cores more evenly when there are many jobs in the system and skew allocations towards more parallelizable jobs when there are few jobs in the system. Unfortunately, GREEDY is not mean field optimal in general (see Appendix I).

Outline. In this section, we develop the GLAM policy, a policy based on a Gittins index style of argument. Our derivation of GLAM shows that the *correct* way to measure the marginal benefit a job receives from each core is the ratio of the reduction in a job's response time to the increase in the job's resource utilization. Using this result, we prove the mean field optimality of GLAM and demonstrate that prior approaches like GREEDY were overly simplistic.

The rest of this section will be devoted to deriving our heuristic policy, GLAM, and then proving that it is asymptotically optimal in the mean field regime. Following a Gittins-style argument (see e.g., [20, 48, 1, 49, 40, 39]), we begin by proposing a relaxation of (2) along two dimensions:

(i) The first relaxation repeats our previous approach of using a time-average constraint on core usage rather than a strict constraint at every time, t (see (5)).

(ii) The second relaxation applies a discount factor relaxation that allows us to put more emphasis on the current allocation and not have to account for everything in the future.

[3] GREEDY is a generalization of the EQUI policy to workloads with multiple classes of jobs

Relaxation (i) above is where we get the primary benefit, as it greatly simplifies the form of the optimization problem. Relaxation (ii) is necessary for technical reasons, and will be resolved at the end when we take a *vanishing discount* limit (similar techniques are used in [36, 31, 49]).

We begin by setting up our Gittins approach in Sections 6.1 and 6.2. In Section 6.3, we derive the GLAM policy. This argument shows how to correctly measure a job's marginal benefit from each additional core, resulting in a policy that adapts allocations based on the number of jobs in the system. Then, in Section 6.4, we show that GLAM is mean field optimal.

6.1. Another Relaxed Optimization Problem

We first alter (2) by applying a discount factor of $e^{-\alpha t}$ to both the objective and constraint. Here, we also re-write the objective function in terms of $N_i(t)$, the total number of class-i jobs in the system at time t. It is easy to see that with $\alpha = 0$, this expression gives the total weighted response time across jobs over a time horizon, τ. For purposes of finding a more nuanced policy, we hold off on collapsing the objective and constraint into per-class expectations.

The constraint is relaxed in an analogous way, but requires tracking $\mathbb{N}_i(t)$, the *set* of class-i jobs in the system at time t. Note that the constraint on total discounted usage becomes $\frac{n}{\alpha}$ as a result of the discounting. The resulting optimization problem is given as follows:

$$\begin{aligned} \underset{k_{ij}(t)}{\text{minimize}} \quad & \lim_{\tau\to\infty} \int_0^\tau e^{-\alpha t} \sum_{i=1}^m N_i(t) c_i dt \\ \text{subject to} \quad & \lim_{\tau\to\infty} \int_0^\tau e^{-\alpha t} \sum_{i=1}^m \sum_{j\in\mathbb{N}_i(t)} k_{ij}(t) dt \le \frac{n}{\alpha} \\ & k_{ij}(t) \in \{0\} \cup [1,\infty) \quad \forall t \ge 0. \end{aligned} \tag{10}$$

6.2. The Dual Problem

Given the optimization problem (10), we can use the method of Lagrange multipliers to set up an auxiliary optimization problem. We begin by defining the Lagrangian as follows. Let $\boldsymbol{k}_\pi(t)$ denote the vector of allocations to all jobs in the system at any time, t, under some scheduling policy, π. Let $f(\boldsymbol{k}_\pi)$ denote the value of the objective function in (10) for a given set of allocations. Furthermore, let $g(\boldsymbol{k}_\pi)$ be the discounted time average core usage under these allocations minus $\frac{n}{\alpha}$. That is, the constraint in (10) is satisfied if $g(\boldsymbol{k}_\pi) \le 0$. We can then write the Lagrangian for this convex constrained optimization problem as follows:

$$\mathcal{L}(\boldsymbol{k}_\pi, \ell) = f(\boldsymbol{k}_\pi) + \ell g(\boldsymbol{k}_\pi).$$

The Lagrange Multiplier Theorem says that, for $\ell \ge 0$, solving (10) is equivalent to solving

$$\min_{\boldsymbol{k}_\pi} \max_{\ell} \mathcal{L}(\boldsymbol{k}_\pi, \ell).$$

Because (10) is convex and strictly feasible (since $\rho < 1$ we can stabilize the system with fewer than n cores), strong Lagrangian duality holds. That is, we can define the Lagrangian dual function,

$$h(\ell) = \min_{\boldsymbol{k}_\pi} \mathcal{L}(\boldsymbol{k}_\pi, \ell).$$

such that, to solve (10), it suffices to solve

$$\max_\ell h(\ell) = \max_\ell \min_{\boldsymbol{k}_\pi} \int_0^\infty e^{-\alpha t} \left[\sum_{i=1}^m N_i(t) c_i + \ell \cdot \sum_{j\in\mathbb{N}_i(t)} k_{ij}(t) \right] dt - \ell \cdot \frac{n}{\alpha}. \tag{11}$$

This expression for the dual problem can then be rewritten and simplified as follows:

$$\max_{\ell} h(\ell) = \max_{\ell} \min_{\boldsymbol{k}_\pi} \int_0^\infty e^{-\alpha t} \left[\sum_{i=1}^{m} \sum_{j \in \mathbb{N}_i(t)} (c_i + \ell \cdot k_{ij}(t)) \right] dt - \ell \cdot \frac{n}{\alpha}. \tag{12}$$

The only dependence between the terms inside the double summation in (12) is captured by the Lagrange multiplier, ℓ. Hence, for a given value of ℓ, the problem of minimizing the inner expression in (12) (and thus yielding the value of $h(\ell)$) decouples into a series of separate optimization problems, one for each job that arrives to the system. Specifically, for the jth job of class i, define an indicator $\mathbb{1}_{ij}(t)$ which is one while the job is active in the system. One must solve the following for each job:

$$\min_{k_{ij}(t)} \int_0^\infty e^{-\alpha t} \mathbb{1}_{ij}(t) \cdot (c_i + \ell \cdot k_{ij}(t))\, dt. \tag{13}$$

We refer to (13) as the *single job problem*.

We then note via complementary slackness that ℓ^*, the value of ℓ that maximizes the Lagrangian dual function, is the value of ℓ where the discounted time average core usage across all jobs is exactly $\frac{n}{\alpha}$. This suggests a two-step optimization process: First, analyze (13) assuming arbitrary values of ℓ. Then, find the value, ℓ^*, such that the total core usage across jobs perfectly uses the available time average budget.

6.3. GLAM: A Gittins-like Policy

Rather than solving (11) via the two-step approach outlined above, we use (11) to derive a heuristic for the original CAM problem based on the approach originally proposed by Gittins [20] and Whittle [48]. We define the *Gittins policy* for the relaxed problem as follows:

Definition 2 (The Gittins policy for discounted problems). At every moment in time, t, the Gittins policy calculates a "market-clearing" service cost, $\widehat{\ell}(t)$. This $\widehat{\ell}(t)$ is defined as the smallest value of ℓ such that, after solving the single job problems defined in (13) for the jobs currently in the system, the total number of cores used across all jobs is no more than n. For each job, the Gittins policy sets $k_{ij}(t)$ to be the number of cores that optimizes (13) given $\ell = \widehat{\ell}(t)$.

For this Gittins policy to be well-defined, there must exist a single $\widehat{\ell}(t)$ satisfying the above conditions. This is generally shown by proving *indexability*, as follows:

Definition 3 (Indexability). Given a discount factor $\alpha > 0$, define the *optimal action* for state x in the single-armed bandit problem (parameterized by discount factor α and service penalty ℓ) to be $k^*(x, \ell)$. The discounted multi-armed bandit problem parameterized by the discount factor α is called *indexable* if for any state x, $k^*(x, \ell)$ is decreasing with ℓ and $\lim_{\ell \to 0} k^*(x, \ell) \to \infty$. That is, when the service cost is lower, the optimal number of cores allocated is higher.

We assume indexability holds in our setting. Prior work [40] proves indexability under very general conditions that are close to our setting but require a finite state space for the underlying problem. We defer formally extending these results to future work. We will only use the assumption of indexability in deriving our Gittins policy. The mean field optimality of our Gittins policy does not rely on indexability – see Theorem 4.

We now derive GLAM (Definition 5), which provides a closed form for the Gittins policy from the previous section. Here, we first derive the Gittins policy for the discounted problem, which does not have a simple closed form. Fortunately, since we truly care about minimizing the undiscounted mean response time, we can take limits as $\alpha \to 0$. This yields a closed form for the Gittins policy in the undiscounted case. While this derivation of GLAM yields a heuristic policy, GLAM turns out to be asymptotically optimal (see Section 6.4) and performs well in practice (see Section 7).

Our central observation about GLAM is that the correct way to measure the marginal benefit of allocating cores to jobs is to look at the *ratio* of response time improvement to increase in resource usage for each additional core. To see this, we define f_i as follows:

Definition 4 (f_i). For any class i job of size x, define

$$f_i(k) := \frac{\text{reduction in response time}}{\text{change in resource usage}} = \frac{-\frac{d}{dk}\left[\frac{x}{s_i(k)}\right]}{\frac{d}{dk}\left[\frac{kx}{s_i(k)}\right]} = \frac{s_i'(k)}{s_i(k) - ks_i'(k)}. \tag{14}$$

Note the independence of the final expression from the job size, x. Hence, f_i is the same for all class-i jobs. Here the derivative of s_i at 1 is defined as $s_i'(1) := \lim_{\epsilon \to 0^+} \frac{s_i(1+\epsilon)-1}{\epsilon}$. Further we define f_i^{-1} to be the inverse function of f_i (the inverse exists because f_i is strictly decreasing, see Lemma 9).

We will show that GLAM allocates cores to the jobs with the highest values of f_i. Theorem 3 will show that, as $\alpha \to 0$, the value of $\widehat{\ell}(t)$ falls into one of two cases. Either

$$\lim_{\alpha \to 0} \widehat{\ell}(t) = \Theta\left(\frac{1}{\alpha}\right) \quad \text{or} \quad \lim_{\alpha \to 0} \widehat{\ell}(t) = \Theta(1).$$

Hence, in Lemma 5 below, it suffices to solve (13) in these two limiting cases.

Lemma 5 (Optimal policy for the single job problem). *The solution to the single job problem in the limit as $\alpha \to 0$ depends on the asymptotic behavior of $\widehat{\ell}(t)$.*
If $\lim_{\alpha \to 0} \widehat{\ell}(t) = \Theta(\frac{1}{\alpha})$, the optimal solution to the single job problem given a job of size x is

$$\begin{cases} \text{Choose action } k = 1 + o(1) & \text{if } \lim_{\alpha \to 0} \widehat{\ell}(t)\alpha < \frac{c_i}{x} \\ \text{Choose action } k = 0 & \text{if } \lim_{\alpha \to 0} \widehat{\ell}(t)\alpha > \frac{c_i}{x}. \end{cases}$$

If $\lim_{\alpha \to 0} \widehat{\ell}(t) = \Theta(1)$, the optimal solution to the single job problem given a job of size x is

$$\begin{cases} \text{Choose action } k = 1 + o(1) & \text{if } \frac{\widehat{\ell}(t)}{c_i} \geq f_i(1) \\ \text{Choose action } k = f_i^{-1}\left(\frac{\widehat{\ell}(t)}{c_i}\right) + o(1) & \text{if } \frac{\widehat{\ell}(t)}{c_i} < f_i(1). \end{cases}$$

Proof. By concavity of the speedup function, the optimal action for the single job problem should not change throughout the job's life. Otherwise, picking the time-average number of cores leads to a better policy (see Lemma 8 in Appendix G for a detailed proof). Hence, we only need to focus on the total cost of each action.

The total discounted cost for action $k = 0$ is

$$\int_0^\infty e^{-\alpha s} c_i ds = \frac{c_i}{\alpha}. \tag{15}$$

For any action $k \geq 1$, the total discounted cost is

$$\begin{aligned} \int_0^{\frac{x}{s_i(k)}} e^{-\alpha s}(c_i + \ell \cdot k) ds = \frac{1}{\alpha}(c_i + \ell k) \cdot \left(1 - e^{-\alpha \frac{x}{s_i(k)}}\right) &= \frac{1}{\alpha}(c_i + \ell k)\left(\alpha \frac{x}{s_i(k)} + o(\alpha)\right) \\ &= \frac{\ell k x}{s_i(k)} + \frac{c_i x}{s_i(k)} + o(1) + o(\ell)k. \end{aligned} \tag{16}$$

We now derive the optimal action for the single job problem for the two cases in Lemma 5.
Case 1: $\ell = \Theta(\frac{1}{\alpha})$. In this case, the comparison between (15) and (16) depends on the sign of $\ell\alpha - \frac{c_i}{x}$.

If $\lim_{\alpha \to 0} \ell\alpha > \frac{c_i}{x}$, we see that (15) (cost when $k = 0$) is smaller than (16) for any $k \geq 1$. Specifically, by sublinearity ($s_i(k) \leq k$), for any $k \geq 1$,

$$\frac{\ell k x}{s_i(k)} + \frac{c_i x}{s_i(k)} + o(1) + o(\ell)\, k \;\geq\; \ell x + O(1) + o(\ell).$$

The condition $\lim_{\alpha\to 0} \ell\alpha > \frac{c_i}{x}$ gives $\ell x - \frac{c_i}{\alpha} = \Theta(\frac{1}{\alpha}) > 0$. Hence, for any $k \geq 1$, (16) is greater than (15) for sufficiently small α. This implies that $k = 0$ is the optimal action as $\alpha \to 0$ in this case.

If $\lim_{\alpha\to 0} \ell\alpha < \frac{c_i}{x}$, we see that $k = 1 + o(1)$ is optimal as $\alpha \to 0$: By (16), the cost of any $k \geq 1$ is $\frac{k}{s_i(k)}\,\ell x + O(1) + o(\ell)k$. At $k = 1$ this is $\ell x + O(1) + o(\ell)$. The assumption $\lim_{\alpha\to 0} \ell\alpha < c_i/x$ now gives $\ell x - \frac{c_i}{\alpha} = \Theta(\frac{-1}{\alpha}) < 0$. Hence, the cost of using $k = 1$ is less than the cost of using $k = 0$ for sufficiently small α. For any $k > 1$, strict sub-linearity of the speedup function gives $\frac{k}{s_i(k)} > 1$, so the cost of using $k > 1$ exceeds the cost of using $k = 1$ by $\Theta(\frac{1}{\alpha})$. This cost difference dominates any differences in the $O(1)$ and $o(\ell)$ terms. Hence the optimal k equals $1 + o(1)$.

Combining the two sub-cases, the optimal action is

$$\begin{cases} \text{Choose action } k = 1 + o(1) & \text{if } \lim_{\alpha\to 0} \ell\alpha < \frac{c_i}{x} \\ \text{Choose action } k = 0 & \text{if } \lim_{\alpha\to 0} \ell\alpha > \frac{c_i}{x}. \end{cases}$$

Case 2: $\ell = \Theta(1)$. Since $\ell = \Theta(1)$, the term $o(\ell)\,k$ in (16) is $o(1)$ (because $k \leq n$). Hence the cost of any $k \geq 1$ is

$$\frac{(\ell k + c_i)\,x}{s_i(k)} + o(1) \;=\; \Theta(1),$$

which is less than the cost $\frac{c_i}{\alpha} = \Theta(\frac{1}{\alpha})$ of $k = 0$. So the optimal action is some $k \geq 1$.

We first minimize the leading part $\frac{(\ell k + c_i)x}{s_i(k)}$, then deal with the $o(1)$ term. Differentiating,

$$\frac{d}{dk}\left[\frac{(\ell k + c_i)x}{s_i(k)}\right] \;=\; \frac{c_i x\,(s_i(k) - k s_i'(k))}{s_i(k)^2}\left(\frac{\ell}{c_i} - f_i(k)\right).$$

By strict concavity of s_i, $s_i(k) - k s_i'(k) > 0$; also f_i is strictly decreasing. Therefore:

- If $\frac{\ell}{c_i} \geq f_i(1)$, then $f_i(k) \leq f_i(1) \leq \frac{\ell}{c_i}$, so the derivative is ≥ 0 throughout, and the leading part is minimized at $k = 1$.
- If $\frac{\ell}{c_i} < f_i(1)$, then there is a unique $k > 1$ where the derivative equals 0, namely $k = f_i^{-1}(\frac{\ell}{c_i})$; the leading part is minimized at this value of k.

Let $\hat{k}$ be the value of k that minimizes this leading term.

We now show that the $o(1)$ term shifts our optimal choice k by at most $o(1)$. Note that by the derivative formula above, the leading part is strictly decreasing before $\hat{k}$ and strictly increasing after. Hence, for any fixed $\epsilon > 0$, the leading part exceeds its minimum by at least some positive constant $c(\epsilon) > 0$ whenever k is at distance at least ϵ from $\hat{k}$. Reduction in the $o(1)$ term in (16) cannot compensate for this $c(\epsilon) = \Theta(1)$ increase as $\alpha \to 0$, so the optimal k is within ϵ of $\hat{k}$. Since ϵ is arbitrary, the optimal k differs from $\hat{k}$ by at most $o(1)$, i.e.,

$$\begin{cases} \text{Choose action } k = 1 + o(1) & \text{if } \frac{\ell}{c_i} \geq f_i(1) \\ \text{Choose action } k = f_i^{-1}\left(\frac{\ell}{c_i}\right) + o(1) & \text{if } \frac{\ell}{c_i} < f_i(1). \end{cases}$$

We are now ready to derive the Gittins policy for scheduling a stream of jobs. Note that a key step in deriving the Gittins policy (Definition 2) is to pick the "market-clearing" service cost $\widehat{\ell}(t)$. In Theorem 3 below, we show that the scaling behavior of $\widehat{\ell}(t)$ depends on the current number of jobs in the system. This allows us to derive a closed form for the Gittins policy as $\alpha \to 0$.

Theorem 3. *Assuming indexability, the Gittins policy in the limit as $\alpha \to 0$ falls into one of two cases at every moment in time:*

1. *If the number of jobs is larger than the number of cores, then $\widehat{\ell}(t)$ scales as $\Theta(1/\alpha)$. The Gittins policy assigns each class-i job an index of $\left(\frac{c_i}{\textit{remaining job size}} + o(1)\right)$. The policy gives $1 + o(1)$ cores to the job with the highest index until insufficient cores remain for the next job.*

2. *Otherwise, $\widehat{\ell}(t)$ scales as $\Theta(1)$ and is chosen such that, if every class-i job gets $g_i(\widehat{\ell}(t))$ cores, the total number of cores used is no more than n. Here $g_i(\ell)$ is defined as*

$$g_i(\ell) := \begin{cases} 1 + o(1) & \text{if } \frac{\ell}{c_i} \geq f_i(1), \\ f_i^{-1}\left(\frac{\ell}{c_i}\right) + o(1) & \text{otherwise,} \end{cases}$$

and where f_i is defined in Definition 4. The Gittins policy allocates every class-i job $g_i(\widehat{\ell}(t))$ cores.

PROOF. By Definition 2, the Gittins policy is determined by selecting the value of ℓ such that the total used cores is equal to n. Now we discuss two cases depending on the number of jobs.

Case 1: number of jobs $\geq$ n. In this case, as $\alpha \to 0$, if $\widehat{\ell}(t) \in \Theta(1)$, by Lemma 5 we know that each job wants at least 1 core, which will exceed our total limit n. Thus, we should make the service cost higher to decrease the total core usage. By Lemma 5, whether a job with weight c and remaining inherent work x wants 1 or 0 cores depends on whether $\widehat{\ell}(t)\alpha > \frac{c}{x} + o(1)$. Hence, $\widehat{\ell}(t)$ must scale as $\Theta\left(\frac{1}{\alpha}\right)$ in order to fulfill the market clearing condition as $\alpha \to 0$.

By Lemma 5, given a value of $\widehat{\ell}(t)$, a job is allocated $1 + o(1)$ cores if and only if $\widehat{\ell}(t)\alpha < \frac{c_i}{x} + o(1)$. Hence, the Gittins policy allocates $1 + o(1)$ cores to each job according to the order of $\frac{c}{x} + o(1)$.

Case 2: number of jobs $<$ n. In this case, because one or more jobs will run on more than 1 core, the market-clearing $\widehat{\ell}(t)$ must scale as $\widehat{\ell}(t) \in \Theta(1)$. Specifically, by Lemma 5, $\widehat{\ell}(t) \in \Theta(1)$ implies that every class-i job gets $1 + o(1)$ cores if $\frac{\widehat{\ell}(t)}{c_i} \geq f_i(1)$, and gets $f_i^{-1}(\frac{\widehat{\ell}(t)}{c_i}) + o(1)$ cores otherwise. Hence, under the market-clearing $\widehat{\ell}(t)$, each job gets $g_i(\widehat{\ell}(t))$ cores and the market-clearing $\widehat{\ell}(t)$ is chosen accordingly.

In both of these cases, a unique market-clearing service cost $\widehat{\ell}(t)$ exists because of indexability.

Finally, we define the GLAM policy, which considers the Gittins policy from Theorem 3 in the limit as $\alpha \to 0$, causing all $o(1)$ and $o(\alpha)$ terms to vanish:

Definition 5 (GLAM). At every moment of time:

- If the number of jobs in the system is at least the number of cores, assign each class-i job an index of $\frac{c_i}{\text{remaining job size}}$. Allocate 1 core to each of the n jobs with the largest index.
- Otherwise, choose the smallest ℓ such that if every class-i job gets $g_i(\ell)$ cores, the total number of cores used is no more than n, where $g_i(\ell)$ is defined as

$$g_i(\ell) := \begin{cases} 1 & \text{if } \frac{\ell}{c_i} \geq f_i(1), \\ f_i^{-1}\left(\frac{\ell}{c_i}\right) & \text{otherwise,} \end{cases}$$

  where f_i and f_i^{-1} are defined in Definition 4. The chosen value of ℓ represents $\widehat{\ell}(t)$, and every class-i job should be allocated $g_i(\widehat{\ell}(t))$ cores.

There are two important intuitions to note about the GLAM policy. First, if we take the classical interpretation of ℓ as a "service charge" assessed to a job for every core it uses, the form of g_i has a nice interpretation. Here, $\frac{\ell}{c_i}$ represents the marginal cost of using an additional core divided by the marginal benefit of reducing the response time of a class-i job by one second. The f_i function similarly looks at the ratio of the marginal change in response time to the marginal change in resource usage. By equating these two ratios, we get

$$\begin{array}{c}\text{change in} \\ \text{response time}\end{array} \times \begin{array}{c}\text{marginal benefit of} \\ \text{response time} \\ \text{reduction}\end{array} = \begin{array}{c}\text{change in} \\ \text{resource} \\ \text{usage}\end{array} \times \begin{array}{c}\text{marginal cost of} \\ \text{resource usage}\end{array}.$$

This matches the classical economic tenet of optimizing profit by setting marginal cost equal to marginal benefit. The GLAM policy then sets a price, $\widehat{\ell}(t)$, such that when each job optimizes its own profit, the resource constraint becomes tight.

Second, one practical way to think about GLAM's allocations is to think about the policy repeatedly allocating one core at a time (or some small fraction of a core) until all cores are allocated. To do this, GLAM roughly awards each core to the job with the highest value of $f_i(k) \cdot c_i$ under the current allocations. By contrast, the GREEDY policy has been described as awarding cores to the jobs with the highest value of $s_i'(k)$. This has a similar flavor to GLAM, but simply uses an overly simplistic heuristic to gauge a job's marginal improvement from each additional core.

6.4. Mean field optimality of GLAM

While we have both a theoretical justification and a strong intuition for GLAM, at this point, it is still just another heuristic policy. Hence, in this section, we prove the mean field optimality of GLAM (Definition 5) subject to some mild conditions. As stated in Section 3, here we assume that job sizes are phase-type distributions. This assumption is mild because phase-type distributions can approximate any distribution arbitrarily closely (Theorem 4.2 in [3]). Without such an assumption, to capture the dynamics of the system in mean field, one would need to use a *measure valued fluid approximation* (e.g., [27]), which is beyond the scope of this paper.

For ease of presentation, we present our proof for the case when job sizes are exponential. Specifically, class-i jobs have size $X_i \sim \text{Exp}(\mu_i)$. The proof for phase type distributions follows a very similar roadmap, and is deferred to Appendix H.

In the mean field regime, the limiting dynamics of the queueing system can be captured by an ordinary differential equation (ODE) [30, 18]. Notably, even for a GI arrival process with rate λ_i, the first-order drift is the same as for a Poisson arrival process with rate λ_i (Theorem 3 in [48]). Thus in the ODE the drift is simply λ_i.

Specifically, we again consider the mean field scaling regime where $d \to \infty$ and ρ remains constant. Let $N_i(t)^{(d)}$ be the number of class-i jobs in the dth system at time t. We can consider the fluid-scaled process $z_i(t) = \frac{N_i(t)^{(d)}}{d}$. According to [18], the GLAM system in the mean field limit can be described by the following ODE:

$$\dot{z}_i = \lambda_i - z_i \cdot \mu_i \cdot s_i(g_i(\ell)), \tag{17}$$

where ℓ is determined by $\sum_{i=1}^{m} z_i \cdot g_i(\ell) = n$ and g_i is defined in Definition 5. Here, λ_i is the arrival rate of class-i jobs, $z_i \cdot \mu_i$ is the total completion rate of class-i jobs (if running on 1 core), and $s_i(g_i(\ell))$ is the speedup that any class-i job gets. Finally, by the definition of GLAM[4], ℓ is picked such that the total number of cores used equals n, which is $\sum_{i=1}^{m} z_i \cdot g_i(\ell) = n$.

We can now state our main theorem. The mean field optimality of GLAM relies on the assumption that ODE (17) has a global attractor. Similar assumptions of the existence of a global attractor are also made in prior works (e.g., [47, 44], see [44] for a detailed discussion)[5].

Theorem 4 (mean field optimality of GLAM). *Assuming ODE* (17) *has a global attractor, the GLAM policy is mean field optimal.*

Proof. Suppose ODE (17) has a global attractor, then the global attractor must be a stationary point. We next prove that there exists a unique stationary point for ODE (17), and the weighted mean response time for the stationary point is equal to the optimal weighted mean response time of the Relaxed problem (see (5)). Note that if this statement holds, we have that GLAM is mean field optimal because its weighted mean response time converges to the lower bound given by the Relaxed problem.

For the stationary point z_i, since $\dot{z}_i = 0$, we have that $z_i \mu_i s_i(g_i(\ell)) = \lambda_i$. Thus we have that

$$z_i = \frac{\lambda_i}{s_i(g_i(\ell))} \mathbf{E}[X_i]. \tag{18}$$

[4] In the mean field limit, the number of jobs in the system is never larger than the number of cores.

[5] Although we conjecture that ODE (17) does have a global attractor, the proof is highly non-trivial even for the case when job sizes are exponential. We leave the verification of this conjecture to future work.

Now we can find the stationary point by solving the equation $\sum_{i=1}^{m} z_i \cdot g_i(\ell) = n$, which is equivalently

$$\sum_{i=1}^{m} \frac{\lambda_i g_i(\ell)}{s_i(g_i(\ell))} \mathbf{E}[X_i] = n. \tag{19}$$

We now argue that there exists a unique $\ell^* < \max_i c_i f_i(1)$ solving (19). First, for any $\ell \geq \max_i c_i f_i(1)$, so we have $g_i(\ell) = 1$ for all i, and the left hand side of (19) becomes $\sum_i \lambda_i \mathbf{E}[X_i] < n$. Next, note that $g_i(\ell)$ is strictly decreasing for any $\ell < c_i f_i(1)$, hence the left hand side of (19) is strictly decreasing with $\ell < \max_i c_i f_i(1)$. Finally, note that as ℓ goes to 0, $g_i(\ell) = f_i^{-1}(\ell/c_i)$ goes to ∞. To see this, note that the numerator of f_i is $o(1)$ by strict sub-linearity and the denominator is $\Omega(1)$ by concavity (specifically the derivative of the denominator is increasing). Hence, there exists a unique ℓ^* solving (19) in the interval $(0, \max_i c_i f_i(1)]$. Therefore, the unique stationary point z_i can be found by plugging ℓ^* into (18).

Since the dynamics of the GLAM system in the mean field limit are captured by an ODE (17) whose global attractor is z_i, we have that in the limit, any class-i job is assigned $g_i(\ell^*)$ cores. Thus the mean response time of a class-i job is $\frac{\mathbf{E}[X_i]}{s_i(g_i(\ell^*))}$, and the weighted mean response time in the limit is

$$\lim_{d\to\infty} \mathbf{E}[T_{\text{Weighted}}]_{CAM^{(d)}}^{GLAM} = \frac{1}{\lambda} \sum_{i=1}^{m} \frac{c_i \lambda_i \mathbf{E}[X_i]}{s_i(g_i(\ell^*))}.$$

One can use Lagrange multipliers to solve the optimization problem (5) and find that the optimal weighted mean response time for the Relaxed problem is $\frac{1}{\lambda} \sum_{i=1}^{m} \frac{c_i \lambda_i \mathbf{E}[X_i]}{s_i(g_i(\ell^*))}$ (Lemma 10). Thus we have that GLAM is mean field optimal.

7. Simulation and Evaluation

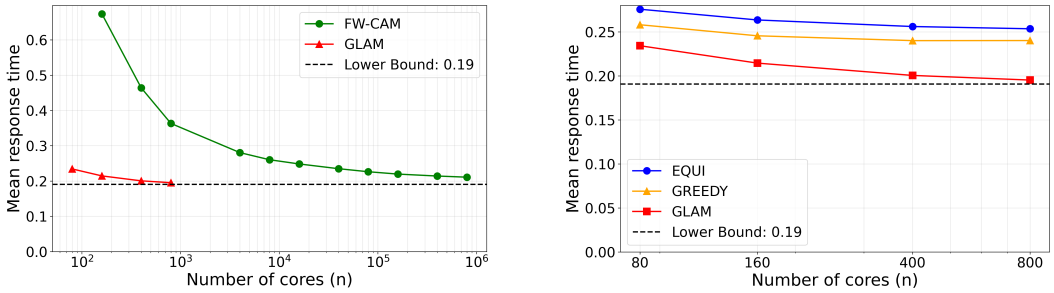
We evaluate the performance of our proposed policies, FW-CAM and GLAM, through numerical simulations. Our evaluation focuses on two aspects: (1) verifying the mean field optimality of our policies and comparing their convergence rates, and (2) demonstrating the superiority of our policies (particularly GLAM) over standard heuristics commonly used in the literature. We ran simulations over a wide range of settings, varying the speedup functions, the system load ρ, and the inter-arrival time and job size distributions. Across all settings the message is the same:

- Both FW-CAM and GLAM converge to the theoretical lower bound given by the Relaxed optimization (5), while standard heuristics from the literature do not.
- GLAM converges significantly faster than FW-CAM and beats the heuristics even when the number of cores is small.

For brevity, the main text reports one *representative* experiment in Figure 3; additional experiments under other inter-arrival time and job size distributions are deferred to Appendix K.

Convergence rates. Figure 3(a) plots the mean response time across jobs as a function of n for each of our policies. Here, we scale $n \to \infty$ in a mean field scaling regime with a fixed load of $\rho = 0.6$. While both of our policies are mean field optimal, GLAM converges significantly faster than FW-CAM, and achieves near-optimal performance even when the number of cores is small.

Comparison to heuristics. We compare GLAM against existing heuristics in the literature. None of these heuristics is mean field optimal, for one of two reasons. (1) The size-based heuristics HELL [37] and (A-)heSRPT [9] prioritize jobs by their remaining size, and are therefore not mean field optimal by Proposition 2 which shows that no mean field optimal policy can depend on exact job sizes in the limit. (2) The size-blind heuristics EQUI [14], KNEE [37], and GREEDY [5] can instead be shown to be strictly suboptimal in mean field via the ODE method of Section 6.4: Appendix I carries out this argument explicitly for GREEDY; similar arguments apply to EQUI and KNEE.



(a) FW-CAM vs. GLAM

(b) GLAM vs. EQUI and GREEDY

Figure 3: Performance evaluation of core allocation policies under a *representative* setting: two classes with sub-linear speedup functions, Poisson arrivals, and Exponential job sizes, with system load fixed at $\rho = \mathbf{0.6}$ as n scales. The same qualitative behavior is observed across all other settings we tested; see Appendix K for additional inter-arrival time and job size distributions and for full workload details.

We implement two representative heuristics in our simulations:

EQUI [14]: A fairness-based policy that, at every moment, divides the total available cores equally among all active jobs in the system.

GREEDY [5]: An efficiency-based policy that, at every moment, maximizes the total rate the system completes inherent work. GREEDY has recently been deployed in real systems (e.g., [41, 26]).

Figure 3(b) compares GLAM against EQUI and GREEDY. The results demonstrate that while GLAM converges to the theoretical optimum, the heuristic policies do not. Furthermore, GLAM consistently outperforms these heuristics for any number of cores, demonstrating its effectiveness even outside of the mean field limit.

8. Conclusion

Prior work on the CAM problem [5, 9] conjectured that optimizing core allocation involves both favoring short jobs and maximizing the total system efficiency. However, general optimality results have remained elusive. By deriving the FW-CAM and GLAM policies, we see that neither of the above intuitions holds in the mean field regime. In particular, our derivation of FW-CAM shows us that any policy that favors short jobs is suboptimal. Likewise, our derivation of the GLAM policy illustrates that maximizing system efficiency, as done by the GREEDY policy, is suboptimal.

This paper derives the first asymptotic optimality results for the CAM problem with multiple concave speedup functions. Our policies FW-CAM and GLAM are asymptotically optimal under very general distributional assumptions on job sizes and inter-arrival times.

References

[1] Samuli Aalto, Urtzi Ayesta, and Rhonda Righter. 2009. On the Gittins index in the M/G/1 queue. *Queueing Systems* 63, 1 (2009), 437.

[2] Samuli Aalto and Ziv Scully. 2023. Minimizing the mean slowdown in the M/G/1 queue. *Queueing Systems* 104, 3 (2023), 187–210.

[3] Søren Asmussen. 2003. *Applied probability and queues*. Springer.

[4] François Baccelli and A Makowski. 1985. *Simple computable bounds for the fork-join queue*. Ph. D. Dissertation. INRIA.

[5] Benjamin Berg, Jan-Pieter Dorsman, and Mor Harchol-Balter. 2017. Towards optimality in parallel scheduling. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–30.

[6] Benjamin Berg and Mor Harchol-Balter. 2021. Optimal scheduling of parallel jobs with unknown service requirements. In *Handbook of Research on Methodologies and Applications of Supercomputing*. IGI Global, 18–40.

[7] Benjamin Berg, Mor Harchol-Balter, Benjamin Moseley, Weina Wang, and Justin Whitehouse. 2020. Optimal resource allocation for elastic and inelastic jobs. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. 75–87.

[8] Benjamin Berg, Benjamin Moseley, Weina Wang, and Mor Harchol-Balter. 2024. Asymptotically optimal scheduling of multiple parallelizable job classes. *arXiv preprint arXiv:2404.00346* (2024).

[9] Benjamin Berg, Rein Vesilo, and Mor Harchol-Balter. 2020. heSRPT: Parallel scheduling to minimize mean slowdown. *Performance Evaluation* 144 (2020), 102–147.

[10] Benjamin Berg, Justin Whitehouse, Benjamin Moseley, Weina Wang, and Mor Harchol-Balter. 2021. The case for phase-aware scheduling of parallelizable jobs. *Performance Evaluation* (2021).

[11] Zhongrui Chen, Isaac Grosof, and Benjamin Berg. 2025. Improving multiresource job scheduling with markovian service rate policies. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 9, 2 (2025), 1–36.

[12] J. G. Dai, Shuangchi He, and Tolga Tezcan. 2010. Many-server diffusion limits for G/Ph/n+GI queues. *The Annals of Applied Probability* 20, 5 (Oct. 2010).

[13] Pamela Delgado, Diego Didona, Florin Dinu, and Willy Zwaenepoel. 2018. Kairos: Preemptive data center scheduling without runtime estimates. In *Proceedings of the ACM Symposium on Cloud Computing*. 135–148.

[14] Jeff Edmonds. 1999. Scheduling in the dark. In *Proceedings of the thirty-first annual ACM symposium on Theory of Computing*. 179–188.

[15] David Gamarnik and David A Goldberg. 2013. Steady-state GI/GI/n queue in the Halfin-Whitt regime. *The Annals of Applied Probability* (2013), 2382–2419.

[16] Nicolas Gast and Gaujal Bruno. 2010. A mean field model of work stealing in large-scale systems. *ACM SIGMETRICS Performance Evaluation Review* 38, 1 (2010), 13–24.

[17] Nicolas Gast, Bruno Gaujal, and Jean-Yves Le Boudec. 2012. Mean field for Markov decision processes: from discrete to continuous optimization. *IEEE Trans. Automat. Control* 57, 9 (2012), 2266–2280.

[18] Nicolas Gast and Benny Van Houdt. 2017. A refined mean field approximation. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 1, 2 (2017), 1–28.

[19] Samira Ghanbarian, Arpan Mukhopadhyay, Ravi R. Mazumdar, and Fabrice M. Guillemin. 2024. On Optimal Server Allocation for Moldable Jobs with Concave Speed-Up. In *Proceedings of the Twenty-Fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing* (Athens, Greece) *(MobiHoc '24)*. Association for Computing Machinery, New York, NY, USA, 191–200.

[20] John C Gittins. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 41, 2 (1979), 148–164.

[21] Isaac Grosof, Ziv Scully, and Mor Harchol-Balter. 2018. SRPT for multiserver systems. *Performance Evaluation* 127-128 (2018), 154–175.

[22] Isaac Grosof, Ziv Scully, Mor Harchol-Balter, and Alan Scheller-Wolf. 2022. Optimal scheduling in the multiserver-job model under heavy traffic. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 3 (2022), 1–32.

[23] Isaac Grosof and Ziyuan Wang. 2024. Bounds on M/G/k scheduling under moderate load improving on SRPT-k and tightening lower bounds. *ACM SIGMETRICS Performance Evaluation Review* 52, 2 (2024), 24–26.

[24] Varun Gupta and Neil Walton. 2019. Load balancing in the nondegenerate slowdown regime. *Operations Research* 67, 1 (2019), 281–294.

[25] Shlomo Halfin and Ward Whitt. 1981. Heavy-traffic limits for queues with many exponential servers. *Operations research* 29, 3 (1981), 567–588.

[26] Suhas Jayaram Subramanya, Daiyaan Arfeen, Shouxu Lin, Aurick Qiao, Zhihao Jia, and Gregory R Ganger. 2023. Sia: Heterogeneity-aware, goodput-optimized ML-cluster scheduling. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 642–657.

[27] Haya Kaspi and Kavita Ramanan. 2011. Law of large numbers limits for many-server queues. (2011).

[28] Cheeha Kim and Ashok K Agrawala. 1989. Analysis of the fork-join queue. *IEEE Transactions on computers* 38, 2 (1989), 250–255.

[29] J. F. C. Kingman. 1962. Some inequalities for the queue GI/G/1. *Biometrika* 49, 3/4 (1962), 315–324.

[30] Thomas G Kurtz. 1970. Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of applied Probability* 7, 1 (1970), 49–58.

[31] Maialen Larrañaga, Urtzi Ayesta, and Ina Maria Verloop. 2014. Index policies for a multi-class queue with convex holding cost and abandonments. In *The 2014 ACM international conference on Measurement and modeling of computer systems*. 125–137.

[32] Guy Latouche and Vaidyanathan Ramaswami. 1999. *Introduction to matrix analytic methods in stochastic modeling*. SIAM.

[33] Viktor Leis, Peter Boncz, Alfons Kemper, and Thomas Neumann. 2014. Morsel-driven parallelism: a NUMA-aware query evaluation framework for the many-core age. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 743–754.

[34] Yuan Li and David A Goldberg. 2025. Simple and explicit bounds for multiserver queues with 1/1- ρ scaling. *Mathematics of Operations Research* 50, 2 (2025), 813–837.

[35] Zhouzi Li, Benjamin Berg, Arpan Mukhopadhyay, and Mor Harchol-Balter. 2024. How to Rent GPUs on a Budget. arXiv:2406.15560

[36] Zhouzi Li, Keerthana Gurushankar, Mor Harchol-Balter, and Alan Scheller-Wolf. 2025. Improving Upon the generalized c-mu rule: a Whittle approach. arXiv:2504.10622

[37] Sung-Han Lin, Marco Paolieri, Cheng-Fu Chou, and Leana Golubchik. 2018. A model-based approach to streamlining distributed training for asynchronous SGD. In *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 306–318.

[38] Randolph Nelson and Asser N Tantawi. 2002. Approximate analysis of fork/join synchronization in parallel queues. *IEEE transactions on computers* 37, 6 (2002), 739–743.

[39] José Niño-Mora. 2001. Restless bandits, partial conservation laws and indexability. *Advances in Applied Probability* 33, 1 (2001), 76–98.

[40] José Niño-Mora. 2022. Multi-gear bandits, partial conservation laws, and indexability. *Mathematics* 10, 14 (2022), 2497.

[41] Aurick Qiao, Sang Keun Choe, Suhas Jayaram Subramanya, Willie Neiswanger, Qirong Ho, Hao Zhang, Gregory R Ganger, and Eric P Xing. 2021. Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning. In *15th USENIX Symposium on Operating Systems Design and Implementation*.

[42] Ziv Scully and Mor Harchol-Balter. 2021. The Gittins policy in the M/G/1 queue. In *2021 19th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 1–8.

[43] Alexey Tumanov, Timothy Zhu, Jun Woo Park, Michael A Kozuch, Mor Harchol-Balter, and Gregory R Ganger. 2016. TetriSched: global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters. In *Proceedings of the Eleventh European Conference on Computer Systems*. 1–16.

[44] I. M. Verloop. 2016. Asymptotically optimal priority policies for indexable and nonindexable restless bandits. *The Annals of Applied Probability* 26, 4 (Aug. 2016).

[45] Benjamin Wagner, André Kohn, and Thomas Neumann. 2021. Self-tuning query scheduling for analytical workloads. In *Proceedings of the 2021 international conference on management of data*. 1879–1891.

[46] Weina Wang, Mor Harchol-Balter, Haotian Jiang, Alan Scheller-Wolf, and Rayadurgam Srikant. 2019. Delay asymptotics and bounds for multi-task parallel jobs. *ACM SIGMETRICS Performance Evaluation Review* 46, 3 (2019), 2–7.

[47] Richard R. Weber and Gideon Weiss. 1990. On an Index Policy for Restless Bandits. *Journal of Applied Probability* 27, 3 (1990), 637–648.

[48] Peter Whittle. 1988. Restless bandits: Activity allocation in a changing world. *Journal of applied probability* 25, A (1988), 287–298.

[49] Peter Whittle. 2005. Tax problems in the undiscounted case. *Journal of Applied Probability* 42, 3 (2005), 754–765.

Appendix A. Convex optimization Translation

In this appendix, we rewrite the optimization problem (5) into a convex optimization problem. For completeness, we state the optimization again (up to some constant in the objective) as below:

$$\begin{aligned} \underset{k_i}{\text{minimize}} \quad & \sum_{i=1}^{m} \frac{\rho_i c_i}{s_i(k_i)} \\ \text{subject to} \quad & \sum_{i=1}^{m} \frac{\rho_i k_i}{s_i(k_i)} \leq 1, \\ & k_i \geq 1. \end{aligned} \tag{A.1}$$

This is not immediately a convex optimization problem because the constraint terms $\frac{k_i}{s_i(k_i)}$ are not necessarily convex. However, by applying a change of variables, we can translate it into a convex optimization problem.

Theorem 5. *The optimization problem* (5) *can be solved in the following two steps:*

1. *Solve the convex optimization problem* (A.2) *to get the optimal solution z_i;*
2. *Get the optimal $k_i = s_i^{-1}(\frac{1}{z_i})$ where s_i^{-1} is the inverse of the speedup function s_i.*

Proof. Define $z_i := \frac{1}{s_i(k_i)}$. Define $d_i := \sup_{k \geq 1} s_i(k)$ (if it does not exist, let $d_i = \infty$). If exists k such that $s_i(k) = d_i$, denote the smallest one by ξ_i; Otherwise, let ξ_i be $\perp$.

Note that since s_i is non-decreasing and concave, we have that it is strictly increasing in $[1, \xi_i]$ ($[1, \infty)$ if $\xi_i = \perp$). Thus we can define the function $\beta_i = s_i^{-1}$ to be the inverse of the speedup function on $[1, d_i]$ ($[1, d_i)$ if $\xi_i = \perp$). Using the fact that s_i is increasing, positive and concave, we have that β_i is increasing and convex.

By definition of z_i, we have that $k_i = \beta_i(\frac{1}{z_i})$. Substituting this into the optimization problem (5), we have

$$\begin{aligned} \underset{z_i}{\text{minimize}} \quad & \sum_{i=1}^{m} \rho_i c_i z_i \\ \text{subject to} \quad & \sum_{i=1}^{m} \rho_i z_i \beta_i(\frac{1}{z_i}) \leq 1 \\ & 1 \geq z_i \geq \frac{1}{d_i}. \end{aligned} \tag{A.2}$$

Note that

$$\frac{d}{dz_i}(z_i \beta_i(\frac{1}{z_i})) = \beta_i(\frac{1}{z_i}) - \frac{1}{z_i}\beta_i'(\frac{1}{z_i}),$$

and

$$\begin{aligned} \frac{d^2}{dz_i^2}(z_i \beta_i(\frac{1}{z_i})) &= \frac{-1}{z_i^2}\beta_i'(\frac{1}{z_i}) + \frac{1}{z_i^2}\beta_i'(\frac{1}{z_i}) + \frac{1}{z_i^2}\beta_i''(\frac{1}{z_i}) \\ &= \frac{1}{z_i^2}\beta_i''(\frac{1}{z_i}) > 0. \end{aligned}$$

Thus the optimization problem (A.2) is a convex optimization, which we can numerically solve for the optimal z_i.

Finally, given the optimal z_i, we can get the optimal k_i by letting $k_i = \beta_i(\frac{1}{z_i})$.

Appendix B. Handling the corner case of infinite optimal k_i

The optimization problem (5) may, in principle, fail to attain its infimum: there exist speedup functions s_i for which sending $k_i \to \infty$ strictly decreases the objective without violating the budget. Such speedups grow nearly linearly in k, in the sense that $\lim_{k\to\infty} k/s_i(k) < \infty$. Realistic speedup functions do not exhibit such linear growth — the marginal benefit of an extra core always diminishes as more cores are added — so this regime arises only as a theoretical corner case. This appendix treats the case for completeness and shows that all results in the main text continue to hold after a single preprocessing step.

Throughout, define

$$L_i := \lim_{k\to\infty} \frac{k}{s_i(k)} \in (0, \infty].$$

High-level approach. The optimization (5) naturally partitions the classes into two groups: a set $\mathcal{F}$ of classes whose optimal allocation is finite ($k_i^* < \infty$), and a set $\mathcal{I}$ of classes whose optimum is attained only as $k_i \to \infty$. Every class in $\mathcal{I}$ necessarily satisfies $L_i < \infty$ — this is what allows $k_i \to \infty$ without violating the budget. The main text analyzes the case $\mathcal{I} = \emptyset$, and this appendix reduces the general case to it: we set aside a dedicated core pool for each $i \in \mathcal{I}$ of just enough size to drive that class's response time to zero in mean field, and apply FW-CAM to the remaining classes $\mathcal{F}$ with a correspondingly reduced budget.

Appendix B.1. Identifying the corner-case classes

We use the change of variables $z_i = 1/s_i(k_i)$ from Appendix A. The optimization (5) is equivalent to the convex program (A.2), with z_i ranging over the compact interval $[0, 1]$ (where $z_i = 0$ corresponds to $k_i = \infty$, and the budget term is extended by continuity to $z_i\beta_i(1/z_i)\big|_{z_i=0} := L_i$).

Since the feasible set in z-coordinates is compact and the constraint $\sum_i \lambda_i \mathbf{E}[X_i]\, z_i\beta_i(1/z_i) \le n$ is strictly convex (proved in Appendix A), the optimization has a *unique* optimum $z^* = (z_i^*)$. Define

$$\mathcal{I} := \{i : z_i^* = 0\}, \qquad \mathcal{F} := \{i : z_i^* > 0\}.$$

By the above, $\mathcal{I}$ and $\mathcal{F}$ are uniquely determined by the problem data. Classes in $\mathcal{I}$ are the corner-case classes; classes in $\mathcal{F}$ have finite optimal allocation $k_i^* = \beta_i(1/z_i^*) < \infty$.

Appendix B.2. Extending FW-CAM to the corner case

We now describe the modification of FW-CAM and verify that it remains mean-field optimal.

Fix any $\beta \in (3/4, 1)$, matching the Sub-Halfin-Whitt scaling used in Appendix F. In the d^{th} system, for each corner-case class $i \in \mathcal{I}$, reserve a dedicated pool of

$$n_i^{(d)} \;:=\; \lambda_i \mathbf{E}[X_i]\, L_i \cdot d \;+\; (\lambda_i \mathbf{E}[X_i]\, L_i \cdot d)^{\beta}$$

cores. Within each pool, class-i jobs are served FCFS, and a job in service is allocated all $n_i^{(d)}$ cores of its pool (so no two class-i jobs ever run concurrently). The remaining $n^{(d)} - \sum_{i\in\mathcal{I}} n_i^{(d)}$ cores are used to run FW-CAM on the restricted class set $\mathcal{F}$, treating the reduced budget as the new total.

We argue that this extension is mean-field optimal by decomposing the weighted mean response time as

$$\mathbf{E}[T_{\text{Weighted}}]^{(d)} = \frac{1}{\lambda}\sum_{i\in\mathcal{I}} \lambda_i c_i \mathbf{E}\left[T_i^{(d)}\right] \;+\; \frac{1}{\lambda}\sum_{i\in\mathcal{F}} \lambda_i c_i \mathbf{E}\left[T_i^{(d)}\right]$$

and treating the two sums separately.

For the $\mathcal{I}$ term, fix $i \in \mathcal{I}$. The class-i pool is a GI/G/1 system: jobs arrive at rate $\lambda_i^{(d)} = d\lambda_i$, and each job of size X_i requires service time $S_i^{(d)} := X_i/s_i(n_i^{(d)})$, so $\mathbf{E}\left[S_i^{(d)}\right] = \mathbf{E}[X_i]/s_i(n_i^{(d)}) = \Theta(d^{-1})$ and $\mathrm{Var}(S_i^{(d)}) = \mathrm{Var}(X_i)/s_i(n_i^{(d)})^2 = \Theta(d^{-2})$. The load is

$$\rho_i^{(d)} \;=\; \lambda_i^{(d)}\, \mathbf{E}\left[S_i^{(d)}\right] \;=\; \frac{d\rho_i}{s_i(n_i^{(d)})} \;=\; \frac{d\rho_i\, L_i}{n_i^{(d)}}\big(1 + o(1)\big) \;=\; \frac{1 + o(1)}{1 + (\rho_i L_i d)^{\beta-1}} \;=\; 1 - \Theta(d^{\beta-1}),$$

where the third equality uses $n_i^{(d)}/s_i(n_i^{(d)}) \to L_i$ as $d \to \infty$, and the fourth substitutes $n_i^{(d)} = \rho_i L_i d + (\rho_i L_i d)^\beta$. So $1 - \rho_i^{(d)} = \Theta(d^{\beta-1})$. By Theorem 2 of [29], the mean queueing time satisfies

$$\mathbf{E}\left[W_q^{(d)}\right] \;\le\; \frac{\lambda_i^{(d)}\big(\mathrm{Var}(A_i^{(d)}) + \mathrm{Var}(S_i^{(d)})\big)}{2(1-\rho_i^{(d)})} \;=\; \Theta(d^{-\beta}) \;\xrightarrow{d\to\infty}\; 0,$$

using $\mathrm{Var}(A_i^{(d)}) = \mathrm{Var}(A_i)/d^2 = \Theta(d^{-2})$. Combined with $\mathbf{E}\left[S_i^{(d)}\right] \to 0$, the class-$i$ mean response time $\mathbf{E}\left[T_i^{(d)}\right] = \mathbf{E}\left[W_q^{(d)}\right] + \mathbf{E}\left[S_i^{(d)}\right]$ vanishes, so the $\mathcal{I}$ term in the decomposition above contributes 0 in the limit.

For the $\mathcal{F}$ term, observe that the reduced budget $n^{(d)} - \sum_{i\in\mathcal{I}} n_i^{(d)} = \big(n - \sum_{i\in\mathcal{I}} \lambda_i \mathbf{E}\left[X_i\right] L_i\big)d - o(d)$ scales linearly in d, and the corresponding optimization on $\mathcal{F}$ has all $k_i^* < \infty$ by construction. Mean-field optimality of FW-CAM on this reduced problem then follows by direct application of the main text.

Finally, in the limit $d \to \infty$, these two contributions together match the infimum of (5) (which, via (4), extends to a lower bound on the CAM problem even in the corner case): the $\mathcal{I}$-classes contribute 0, and the $\mathcal{F}$-classes contribute exactly the optimal value of the reduced optimization. Hence the extended policy is mean-field optimal.

Appendix C. Almost no parallelism happens in asymptotic regimes that $\rho \to 1$

The following proposition shows that for the CAM problem with strictly sub-linear speedup functions under asymptotic regimes where $\rho \to 1$, the fraction of work done in parallelism goes to 0. Most notation is defined in Section 3.

Proposition 1. *Define $\rho^{(d)} := \sum_{i=1}^m \lambda_i^{(d)} \mathbf{E}\left[X_i\right]/n^{(d)}$ to be the total load in the d^{th} system. Assume that the speedup functions of all classes are strictly sub-linear and the asymptotic regime has $\lim_{d\to\infty} \rho^{(d)} \to 1$. For any $k_0 > 1$, further define $q^{(d)}(k_0)$ to be the time-average proportion of inherent work finished using more than k_0 cores. Then*

$$\forall k_0 > 1,\ \lim_{d\to\infty} q^{(d)}(k_0) \to 0.$$

Proof. Recall that each type-i job brings X_i inherent work to the system. Thus we have that the time average inflowing rate of the inherent work is

$$\sum_{i=1}^m \lambda_i^{(d)} \cdot \mathbf{E}\left[X_i\right] = \rho^{(d)} \cdot n^{(d)}. \tag{C.1}$$

If a class-i job is running on $k \ge k_0$ cores, then a unit inherent work is finished in $\frac{1}{s_i(k)}$ units of time. Thus the *core-second* needed for a unit inherent work is $k \cdot \frac{1}{s_i(k)} > \frac{k_0}{s_i(k_0)}$ (because of the concavity of the speedup function s_i).

Thus, for the $q^{(d)}(k_0)$ proportion of inherent work finished using more than k_0 cores, the total core-second needed is at least

$$q^{(d)}(k_0) \cdot \rho^{(d)} n^{(d)} \cdot \frac{k_0}{\max_i s_i(k_0)}.$$

Moreover, for the rest $1 - q^{(d)}(k_0)$ proportion of inherent work, by sub-linearity of the speedup functions, the total core-second needed is at least $(1 - q^{(d)}(k_0)) \cdot \rho^{(d)} n^{(d)}$. Note that for the system to be stable, the time average total core-second needed must be no more than $n^{(d)}$. Thus we have

$$q^{(d)}(k_0) \cdot \rho^{(d)} n^{(d)} \cdot \frac{k_0}{s_i(k_0)} + (1 - q^{(d)}(k_0)) \cdot \rho^{(d)} n^{(d)} \le n^{(d)}.$$

This inequality gives that

$$q^{(d)}(k_0) \leq \left(\frac{1}{\rho^{(d)}} - 1\right) \cdot \left(\frac{1}{\frac{k_0}{\max_i s_i(k_0)} - 1}\right).$$

Since $\lim_{d\to\infty} \rho^{(d)} \to 1$, we have that for any $k_0 > 1$, $\lim_{d\to\infty} q^{(d)}(k_0) \to 0$.

Appendix D. Mean field optimal policies should not use exact job size information

In this section, we prove that for any policy that is mean field optimal, it almost surely allocates the same number of cores to all class-i jobs, regardless of their exact job size. The proof uses results and notation from Section 4.

Proposition 2. *Assume the speedup function for any class i is strictly concave. Define $k_i := k_i^*(\lambda_i, n)$. For any policy π that is mean field optimal, and for any class i, we have that the fraction of class-i inherent work completed using k_i cores goes to 1 in mean field.*

Proof. We prove the proposition by contradiction. Suppose the statement does not hold, then either (1) the fraction of class-i work completed using $k' \neq k_i$ goes to 1 in mean field, or (2) the fraction of class-i work is distributed among multiple values (or a range) of k, such that the fraction does not converge to 1 for any single core count.

We first prove that (2) cannot hold for any class i. Suppose (2) holds for some class i. Let $h_i(k)$ be the fraction of time that a class-i job is running on k cores ($\int_1^\infty h_i(k)dk = 1$). Now we create a policy π' for the Relaxed problem, which always allocates $k' := \int_1^\infty k \cdot h_i(k)dk$ cores to class-i jobs. By concavity of the speedup function, we have that the mean response time of class-i job under π' is no worse than that under π (see Lemma 1 in [35]). Moreover, the time-average number of cores used under π' is strictly better than that under π: the time-average number of cores used under π' is $\frac{k'\lambda_i \mathbf{E}[X_i]}{s_i(k')}$, and that under π is

$$\begin{aligned}
\int_0^{k'} \frac{k\lambda_i \mathbf{E}[X_i] \cdot \frac{s_i(k)h_i(k)}{\int_1^\infty s_i(x)h_i(x)dx}}{s_i(k)} dk &= \int_0^{k'} k\lambda_i \mathbf{E}[X_i] \cdot \frac{h_i(k)}{\int_1^\infty s_i(x)h_i(x)dx} dk \\
&= \frac{1}{\int_1^\infty s_i(x)h_i(x)dx} \lambda_i \mathbf{E}[X_i] k' \\
&> \frac{\lambda_i \mathbf{E}[X_i] k'}{s_i(k')}.
\end{aligned}$$

Define the reduced ratio of the time-average number of cores used to be some $c > 0$. Then, we have that

$$\begin{aligned}
\lim_{d\to\infty} \mathbf{E}[T_{\text{Weighted}}]^{\pi}_{CAM^{(d)}} &= \lim_{d\to\infty} \mathbf{E}[T_{\text{Weighted}}]^{\pi}_{Relaxed^{(d)}} \\
&\geq \lim_{d\to\infty} \mathbf{E}[T_{\text{Weighted}}]^{\pi'}_{Relaxed^{(d)}}.
\end{aligned}$$

Note that π' only uses $(1-c)n$ cores on average in the Relaxed problem. By (5), the optimal weighted mean response time is strictly larger than the optimal weighted mean response time of the Relaxed policy using on average n cores because the resource constraint shrank by a $(1-c)$ factor. Hence the weighted mean response time under π is strictly larger than that of the optimal Relaxed problem, hence is sub-optimal.

Next, since (2) does not hold for any class i, under π each class i job only runs on a single number of cores in mean field, denoted by k_i'. This means that k_i' is a feasible solution of the optimization problem (5). Since the speedup functions are strictly concave, the optimization problem 5 can be translated into a strictly convex optimization and the optimal solution is unique. Thus, $k_i' = k_i$.

Appendix E. Proof for Lemma 1

In this section, we prove Lemma 1, which is listed below.

Lemma 6. *For any class i,* $\lim_{d\to\infty} k_i^*\left(\lambda^{(d)}, n^{(d)} - \left(n^{(d)}\right)^{\beta}\right) \to k_i^*(\lambda, n)$.

Proof. By definition, $k_i^*\left(\lambda^{(d)}, n^{(d)} - \left(n^{(d)}\right)^{\beta}\right)$ is the optimal solution of the following optimization problem:

$$\begin{aligned} \underset{k_i}{\text{minimize}} \quad & \sum_{i=1}^{m} \frac{\lambda_i \cdot d}{\lambda \cdot d} \frac{\mathbf{E}[X_i]\, c_i}{s_i(k_i)} \\ \text{subject to} \quad & \sum_{i=1}^{m} \frac{\lambda_i \cdot d \mathbf{E}[X_i]\, k_i}{s_i(k_i)} \leq n \cdot d - (n \cdot d)^{\beta}. \end{aligned} \tag{E.1}$$

Dividing by d on both sides of the constraint, we have that the optimization problem is equivalent to:

$$\begin{aligned} \underset{k_i}{\text{minimize}} \quad & \sum_{i=1}^{m} \frac{\lambda_i}{\lambda} \frac{\mathbf{E}[X_i]\, c_i}{s_i(k_i)} \\ \text{subject to} \quad & \sum_{i=1}^{m} \frac{\lambda_i \mathbf{E}[X_i]\, k_i}{s_i(k_i)} \leq n - n^{\beta} \cdot d^{\beta-1}. \end{aligned} \tag{E.2}$$

Note that both the objective function and constraint function are continuous in k_i. Moreover, the optimization problem is a convex optimization problem (see Appendix A). Thus, since $\lim_{d\to\infty} d^{\beta-1} \to 0$, we have that both the solution and the value of the optimization problem (E.2) converge to those of the optimization problem (5).

Appendix F. Verification of the conditions in [34]

Here, we state Corollary 3 from [34] in its original form.

Lemma 7 (From Corollary 3 in [34]). *For a GI/GI/n queue with inter-arrival time A and job size distribution S, suppose:* (i) $\mathbf{E}\left[A^2\right] < \infty$, (ii) $\mathbf{E}\left[S^3\right] < \infty$, (iii) $\frac{1}{\mathbf{E}[A]} < n\frac{1}{\mathbf{E}[S]}$, (iv) $n(1-\rho)^2 \geq 10^6 \left(\mathbf{E}\left[(\frac{A}{\mathbf{E}[A]})^2\right]\right)^2$, *and* (v) *a steady-state distribution of number of jobs in the system exists, then*

$$\mathbf{E}[\textit{steady-state number of queueing jobs}] \leq \frac{a_1 + a_2}{1-\rho} \cdot 1000(n(1-\rho)^2)^{-0.5},$$

where a_1, a_2 are constant with respect to n.

To apply Lemma 7, we will show that the conditions for the Lemma are satisfied in a GI/GI/$n_i'^{(d)}$ system for sufficiently large d. Specifically, we note that condition (iv) in the Lemma will be satisfied for sufficiently large d if the system is in a Sub-Halfin-Whitt scaling regime. Furthermore, the bound on the number of jobs in queue will converge to 0 if the system is in a strongly Sub-Halfin-Whitt regime where the system load scales as $1 - \omega\left(\left(n^{(d)}\right)^{-.25}\right)$. For our purposes in this paper, this understanding of the Corollary is sufficient to prove the optimality of FW-CAM. However, [34] states the result in the above, more general form to give additional information about convergence rates and scaling behavior. We now verify the conditions for the Lemma below.

As is also assumed in [34], we assume that the steady-state distribution of the number of jobs in system exists. Hence, (v) is satisfied by assumption. We now verify that conditions (i) - (iv) hold for sufficiently large d.

(i) $\mathbf{E}\left[A_i^2\right] < \infty$: In the d^{th} system, we have that $\mathbf{E}\left[(A_i^{(d)})^2\right] = \frac{\mathbf{E}[A_i^2]}{d^2}$, which is finite for any d.

(ii) $\mathbf{E}\left[S^3\right] < \infty$: In the d^{th} system, we have that $\mathbf{E}\left[\left(S_i^{(d)}\right)^3\right] = \mathbf{E}\left[\left(\frac{X_i}{s_i(k_i^{(d)})}\right)^3\right]$, which is finite for any d.

(iii) $\frac{1}{\mathbf{E}[A]} < n\frac{1}{\mathbf{E}[S]}$: In the d^{th} system, define $\rho_i^{(d)} := \frac{\lambda_i^{(d)}\mathbf{E}\left[S_i^{(d)}\right]}{\left(n_i'\right)^{(d)}}$. Then, this inequality is equivalent to $\rho_i^{(d)} < 1$, and it suffices to verify the inequality (F.1) below.

(iv) $n(1-\rho)^2 \geq 10^6\left(\mathbf{E}\left[(\frac{A}{\mathbf{E}[A]})^2\right]\right)^2$: In the d^{th} system, this inequality is equivalent to

$$\left(n_i'\right)^{(d)}\left(1-\rho_i^{(d)}\right)^2 \geq 10^6\left(\frac{\mathbf{E}\left[\left(A_i^{(d)}\right)^2\right]}{\mathbf{E}\left[A_i^{(d)}\right]^2}\right)^2. \tag{F.1}$$

Since we only care about systems when $d \to \infty$, we can simplify the terms by considering their orders with respect to d. By definition, we have $\lambda_i^{(d)} = \Theta(d)$. By Lemma 1, we have that $\mathbf{E}\left[S_i^{(d)}\right] = \Theta(1)$ and $k_i^{(d)} = \Theta(1)$. Thus we have that

$$\begin{aligned}
\left(n_i'\right)^{(d)} &= \lfloor n^{(d)} \cdot \frac{r_i^{(d)}}{\sum_{j=1}^m r_j^{(d)}} \cdot \frac{1}{k_i^{(d)}} \rfloor \\
&= n^{(d)} \cdot \frac{r_i^{(d)}}{\sum_{j=1}^m r_j^{(d)}} \cdot \frac{1}{k_i^{(d)}} + O(1) \\
&= n^{(d)} \cdot \frac{\lambda_i^{(d)}\mathbf{E}\left[S_i^{(d)}\right]}{\sum_{j=1}^m r_j^{(d)}} + O(1) \\
&\geq \lambda_i^{(d)}\mathbf{E}\left[S_i^{(d)}\right] \cdot \frac{n^{(d)}}{n^{(d)} - \left(n^{(d)}\right)^{\beta}} + O(1) \qquad \text{by (7)} \qquad \text{(F.2)} \\
&= \Theta(d). \qquad \text{(F.3)}
\end{aligned}$$

Moreover, we have that $\left(n_i'\right)^{(d)} \leq n^{(d)} = \Theta(d)$. Hence,

$$\left(n_i'\right)^{(d)} = \Theta(d). \tag{F.4}$$

We can now analyze the order of $1-\rho_i^{(d)}$:

$$
\begin{aligned}
1-\rho_i^{(d)} &= \frac{(n_i')^{(d)} - \lambda_i^{(d)}\mathbf{E}\left[S_i^{(d)}\right]}{(n_i')^{(d)}} \\
&\geq \frac{\lambda_i^{(d)}\mathbf{E}\left[S_i^{(d)}\right]\cdot\frac{n^{(d)}}{n^{(d)}-\left(n^{(d)}\right)^{\beta}} + O(1) - \lambda_i^{(d)}\mathbf{E}\left[S_i^{(d)}\right]}{\Theta(d)} \\
&= \frac{1}{\Theta(d)}\left(\lambda_i^{(d)}\mathbf{E}\left[S_i^{(d)}\right]\frac{\left(n^{(d)}\right)^{\beta}}{n^{(d)}-\left(n^{(d)}\right)^{\beta}} + O(1)\right) \\
&= \frac{1}{\Theta(d)}\left(\Theta(d)\frac{\Theta(d^{\beta})}{\Theta(d)} + O(1)\right) \\
&= \Theta(d^{\beta-1})
\end{aligned}
$$

Thus, the left hand side of inequality (F.1) is $\Theta(d^{2\beta-1})$ where $2\beta-1>0$, while the right hand side of (F.1) is of order $\Theta(1)$ by definition. Hence the inequality always holds when $d\to\infty$.

Appendix G. Omitted proof for optimal policy of the single-job problem

In this appendix we prove the following omitted lemma in the proof of Lemma 5.

Lemma 8. *For any discount factor $\alpha\to 0$ and service penalty ℓ, the optimal policy solving* (13) *always chooses the same action throughout the job's lifetime. In other words, the optimal policy either always chooses action $k=0$ and leaves the job forever incurring some cost, or keeps choosing some action $k\geq 1$ until the job is completed.*

Proof. First, if the optimal action at state x is to choose $k=0$, since the state will not change, we have that the optimal policy will choose $k=0$ forever.

Otherwise, the lemma holds by concavity of the speedup function s_i. Mathematically, suppose the optimal policy π chooses action $k(t)\geq 1$ at time t for $t\in[0,t']$. One can create another policy π', which use the same action throughout this t' time, incurring the same service penalty but does more work. Mathematically, define

$$
k' = \frac{\alpha}{1-e^{-\alpha t'}}\int_0^{t'} k(t)e^{-\alpha t}dt.
$$

The policy π' allocates k' cores to the job throughout time $[0,t']$. Compared with π, the two policies incur exactly the same total discounted service penalty for any α and ℓ. However, the total work π' does is $s_i(k')t'$, while the total work π does is $\int_0^{t'} s_i(k(t))dt$. Note that when $\alpha\to 0$, k' goes to the time-average value of $k(t)$ within time $[0,t']$. Thus, by strict concavity of the speedup function, we have that π' is strictly better than π when $\alpha\to 0$.

We also prove the following lemma.

Lemma 9. *For any class i, the function $f_i(k) := \frac{s_i'(k)}{s_i(k)-ks_i'(k)}$ is strictly decreasing with k.*

Proof. We only need to show that $\frac{1}{f_i(k)}$ is increasing, which is $\frac{s_i(k)}{s_i'(k)}-k$.

We can take the derivative of this expression:

$$
\begin{aligned}
\frac{d(\frac{1}{f_i(k)})}{dk} &= \frac{s_i'(k)^2 - s_i(k)s_i''(k)}{s_i'(k)^2} - 1 \\
&= \frac{-s_i(k)s_i''(k)}{s_i'(k)^2} \\
&> 0.
\end{aligned}
$$

The last inequality holds because the speedup functions are (strictly) concave.

Appendix H. Proof for job sizes that follow phase-type distribution

In this appendix, we present the proof for the mean field optimality of GLAM under the assumption of phase-type distributed job sizes. The proof is similar to that presented in Section 6.4.

Assumption 1. *Assume the job size distribution X_i is a phase-type distribution defined by the following Markov chain: There are K_i states. Let $\boldsymbol{p}_i = (p_{iu})$ denote the vector of the initial probability to be in state u, and let $\boldsymbol{Q}_i$ be the sub-generator matrix of the Markov chain, i.e., $\boldsymbol{Q}_i$ is a $K_i \times K_i$ matrix, and $\boldsymbol{Q}_i(u,v)$ is the transition rate from u to v $u \neq v$, $\boldsymbol{Q}_i(u,u) := -\sum_{v \neq u} \boldsymbol{Q}_i(u,v) - q_i(u,0)$ where $q_i(u,0)$ is the transition rate from state u to the absorbing state.*

Under the mean field regime, the limiting dynamics of the queueing system can be captured by an ODE [30, 18]. Notably, even for a GI arrival process, the first-order drift is the same as Poisson arrivals, thus it can be treated as if it is a Poisson arrival in the ODE (Theorem 3 in [48]). Thus, the system under policy GLAM in mean field can be described by the following ODE:

$$\dot{\mathbf{z}}_i = \lambda_i \mathbf{p}_i + \mathbf{z}_i \mathbf{Q}_i \cdot s_i(g_i(\ell)), \tag{H.1}$$
$$\text{where } \ell \text{ is determined by } \sum_{i=1}^{m} \mathbf{z}_i \cdot \mathbf{1} \cdot g_i(\ell) = n, \text{ and } g_i \text{ is defined in Definition 5.}$$

Here $\mathbf{z}_i$ is the vector of the fluid-scaled number of class-i jobs, and $\mathbf{z}_i(u)$ is the fluid-scaled number of class-i jobs in state u. For a better understanding of the ODE (H.1), $\lambda_i \mathbf{p}_i$ is the arrival rate of class-i jobs, $\mathbf{z}_i \mathbf{Q}_i$ represents the rate of transition between the states of class-i jobs (if running on 1 core), and $s_i(g_i(\ell))$ is the speedup that any class-i job gets. Finally, by the definition of GLAM, ℓ is picked such that the total number of cores used reaches n, which is $\sum_{i=1}^{m} \mathbf{z}_i \cdot \mathbf{1} \cdot g_i(\ell) = n$.

We can now state our main theorem. Note that the asymptotic optimality of GLAM relies on the assumption that ODE (H.1) has a global attractor. Similar assumptions of the existence of a global attractor are also made in prior works (e.g., [47, 44], see [44] for a detailed discussion).

Theorem 6 (mean field optimality of GLAM). *If ODE* (H.1) *has a global attractor, then GLAM is mean field optimal.*

Proof. Suppose ODE (H.1) has a global attractor, then the global attractor must be a stationary point. We next prove that there exists a unique stationary point for ODE (H.1), and the weighted mean response time for the stationary point is equal to the optimal weighted mean response time of the Relaxed problem (see (5)). Note that if this statement holds, we have that GLAM is mean field optimal because its weighted mean response time converges to the lower bound given by the Relaxed problem.

For the stationary point $\mathbf{z}_i$, since $\dot{\mathbf{z}}_i = 0$, we have that

$$\mathbf{z}_i \mathbf{Q}_i \cdot s_i(g_i(\ell)) = -\lambda_i \mathbf{p}_i.$$

Since matrix $\mathbf{Q}_i$ is invertible (Lemma 2.2.1 in [32]), we have that

$$\mathbf{z}_i = -\frac{\lambda_i}{s_i(g_i(\ell))} \mathbf{p}_i \left(\mathbf{Q}_i\right)^{-1}. \tag{H.2}$$

Moreover, we have that $\mathbf{E}[X_i] = -\mathbf{p}_i \left(\mathbf{Q}_i\right)^{-1} \mathbf{1}$ (Theorem 2.2.3 in [32]). Thus, by multiplying $\mathbf{1}$ on both sides of (H.2), we have that

$$\mathbf{z}_i \mathbf{1} = \frac{\lambda_i}{s_i(g_i(\ell))} \mathbf{E}[X_i].$$

Now we can find the stationary point by solving the equation

$$\sum_{i=1}^{m} \mathbf{z}_i \mathbf{1} \cdot g_i(\ell) = n,$$

which is equivalently

$$\sum_{i=1}^{m} \frac{\lambda_i g_i(\ell)}{s_i(g_i(\ell))} \mathbf{E}[X_i] = n. \tag{H.3}$$

Note that g_i is strictly decreasing for any $\ell < c_i f_i(1)$, also $\frac{k}{s_i(k)}$ is strictly increasing with k, thus we have that the left hand side is strictly decreasing with $\ell < \max_i c_i f_i(1)$. Moreover, for any $\ell \geq \max_i c_i f_i(1)$, we have $g_i(\ell) = 1$ for any i, and the left hand side is equal to $\sum_i \lambda_i \mathbf{E}[X_i] < n$. This means there exists a unique ℓ^* solving (H.3). Then, the stationary point $\mathbf{z}_i$ can be solved from (H.2).

Since the dynamics of the system under GLAM in mean field is captured by ODE (H.1) whose global attractor is $\mathbf{z}_i$, we have that in the limit, any class-i job is assigned $g_i(\ell^*)$ cores. Thus the mean response time of a class-i job is $\frac{\mathbf{E}[X_i]}{s_i(g_i(\ell^*))}$, and the weighted mean response time in the limit is

$$\lim_{d\to\infty} \mathbf{E}[T_{\text{Weighted}}]^{GLAM}_{CAM^{(d)}} = \frac{1}{\lambda} \sum_{i=1}^{m} \frac{c_i \lambda_i \mathbf{E}[X_i]}{s_i(g_i(\ell^*))}.$$

One can use Lagrange multipliers to solve the optimization problem (5) and find that the optimal weighted mean response time for the Relaxed problem is $\frac{1}{\lambda} \sum_{i=1}^{m} \frac{c_i \lambda_i \mathbf{E}[X_i]}{s_i(g_i(\ell^*))}$ (Lemma 10). Thus we have that GLAM is mean field optimal.

Appendix I. GREEDY is not mean field optimal

In Section 6 and Section 7, we argued informally that GREEDY [5] fails to be mean field optimal. In this appendix, we make this precise via the ODE method of Section 6.4: we write down the mean field ODE for GREEDY, identify its stationary point, and observe that it differs from the optimum identified in Theorem 4. Similar methods apply to other heuristics in the literature, such as EQUI [14] and KNEE [37]; we focus on GREEDY for concreteness.

We treat the case of exponential job sizes ($X_i \sim \text{Exp}(\mu_i)$) and unit weights ($c_i = 1$); since GREEDY was designed for the mean response time, this case alone is enough to exhibit its suboptimality.

Appendix I.1. GREEDY's allocation in mean field

In any state with N_i class-i jobs in the system, GREEDY gives every class-i job the same allocation k_i (by symmetry and the concavity of s_i), where $\{k_i\}$ solve

$$\max \sum_{i=1}^{m} N_i \, s_i(k_i) \quad \text{subject to} \quad \sum_{i=1}^{m} N_i k_i \leq n.$$

A standard marginal argument shows that the resulting per-job allocation is

$$h_i(\eta) \;:=\; \begin{cases} 1 & \text{if } s_i'(1) \leq \eta, \\ (s_i')^{-1}(\eta) & \text{otherwise,} \end{cases}$$

where η is the smallest "marginal value of a core" such that the total allocation fits the budget. So GREEDY is itself a "market-clearing" policy, but it equalizes $s_i'(k_i)$ across classes, whereas GLAM (Definition 5, under $c_i = 1$) equalizes $f_i(k_i)$. Since these two equalizing conditions differ, GREEDY's per-class allocations differ from GLAM's, and as we now show this discrepancy means GREEDY's ODE stationary point is not the mean field optimum.

Appendix I.2. The ODE and its stationary point

In the mean field scaling regime ($d \to \infty$, ρ fixed), let $z_i(t)$ be the fluid-scaled number of class-i jobs in the system. By the standard fluid limit [30, 18] (using Theorem 3 of [48] for GI arrivals), GREEDY's mean field dynamics are

$$\dot{z}_i \;=\; \lambda_i \;-\; z_i\,\mu_i\,s_i\big(h_i(\eta)\big), \qquad \text{where } \eta \text{ satisfies} \quad \sum_{i=1}^{m} z_i\,h_i(\eta) = n. \tag{I.1}$$

This is exactly the GLAM ODE (17) with $g_i(\ell)$ replaced by $h_i(\eta)$.

Setting $\dot{z}_i = 0$ yields the stationary point

$$z_i^{\mathrm{GREEDY}} \;=\; \frac{\lambda_i\,\mathbf{E}\left[X_i\right]}{s_i\big(h_i(\eta^{\mathrm{GREEDY}})\big)}, \qquad \text{where } \eta^{\mathrm{GREEDY}} \text{ solves} \quad \sum_{i=1}^{m} \frac{\lambda_i\,\mathbf{E}\left[X_i\right]\,h_i(\eta^{\mathrm{GREEDY}})}{s_i\big(h_i(\eta^{\mathrm{GREEDY}})\big)} \;=\; n. \tag{I.2}$$

By the same monotonicity argument as in the proof of Theorem 4, η^{GREEDY} exists and is unique. Assuming it is a global attractor of (I.1) (in analogy with the assumption made for GLAM), the mean response time of GREEDY in mean field is

$$\lim_{d\to\infty} \mathbf{E}\left[T_{\mathrm{Weighted}}\right]^{\mathrm{GREEDY}}_{CAM^{(d)}} \;=\; \frac{1}{\lambda}\sum_{i=1}^{m} \frac{\lambda_i\,\mathbf{E}\left[X_i\right]}{s_i\big(h_i(\eta^{\mathrm{GREEDY}})\big)}. \tag{I.3}$$

Appendix I.3. Why GREEDY can be suboptimal in mean field

The expressions (I.2)–(I.3) mirror those for GLAM (equations (18)–(19) and the cost expression in Theorem 4); only the per-class allocations differ. By Lemma 10, the lower bound from the Relaxed problem (5) is achieved exactly when every class-i job gets $g_i(\ell^*)$ cores, and this minimizer is unique by the strict convexity of (5) (Appendix A). Hence GREEDY matches this lower bound only when $h_i(\eta^{\mathrm{GREEDY}}) = g_i(\ell^*)$ for every class. However, the GREEDY allocations $h_i(\eta^{\mathrm{GREEDY}})$ do not agree with the GLAM allocations $g_i(\ell^*)$ in most cases. For example, in the two-class workload evaluated in Section 7, the two allocations disagree and thus GREEDY is suboptimal in mean field, matching the gap seen in Figure 3(b) and Figure K.4.

Appendix J. Solution of the optimization problem 5

Lemma 10. *We have that the optimal value of the optimization* (5) *is given by*

$$\frac{1}{\lambda}\sum_{i=1}^{m} \frac{c_i\lambda_i\mathbf{E}\left[X_i\right]}{s_i(g_i(\ell^*))},$$

where ℓ^* is defined by $\sum_{i=1}^{m} \frac{\lambda_i g_i(\ell^*)}{s_i(g_i(\ell^*))}\mathbf{E}\left[X_i\right] = n$.

Proof. Using Lagrange multiplier method, we define

$$L := \sum_{i=1}^{m} \frac{\lambda_i\mathbf{E}\left[X_i\right]c_i}{s_i(k_i)} + \gamma\left(\cdot\sum_{i=1}^{m} \frac{\lambda_i\mathbf{E}\left[X_i\right]k_i}{s_i(k_i)} - n\right).$$

Then by the KKT condition, we have that the optimal policy k_i satisfies:

$$\frac{\partial L}{\partial k_i} = 0 \quad \text{or} \quad \big(k_i = 1 \text{ and } \frac{\partial L}{\partial k_i}\Big|_{k_i=1} \geq 0\big)$$
$$\sum_{i=1}^{m} \frac{\lambda_i\mathbf{E}\left[X_i\right]k_i}{s_i(k_i)} = n.$$

Note that

$$\frac{\partial L}{\partial k_i} = \lambda_i \mathbf{E}[X_i] \left(\frac{c_i \cdot (-1) s_i'(k_i)}{s_i(k_i)^2} + \gamma \frac{s_i(k_i) - k_i s_i'(k_i)}{s_i(k_i)^2} \right).$$

Thus, $\frac{\partial L}{\partial k_i} = 0$ is equivalent to

$$\frac{\gamma}{c_i} = \frac{s_i'(k_i)}{s_i(k_i) - k_i s_i'(k_i)} = f_i(k_i),$$

and $k_i = 1$ if and only if $f_i(1) \leq \frac{\gamma}{c_i}$. Thus we have $k_i = g_i(\gamma)$.

Now since both ℓ^* and γ are solutions of the equation $\sum_{i=1}^{m} \frac{\lambda_i g_i(\ell)}{s_i(g_i(\ell))} \mathbf{E}[X_i] = n$, which only has a unique solution, we have that $\gamma = \ell^*$. Thus the optimal $k_i = g_i(\ell^*)$, and the optimal value of the optimization problem (5) is

$$\frac{1}{\lambda} \sum_{i=1}^{m} \frac{c_i \lambda_i \mathbf{E}[X_i]}{s_i(g_i(\ell^*))}.$$

Appendix K. Simulation Details

This appendix provides the full workload specification for Section 7 and reports additional experiments that vary the inter-arrival time and job size distributions. As discussed in Section 7, the qualitative message is the same across all settings we tested: FW-CAM and GLAM converge to the lower bound, the heuristics do not, and GLAM beats the heuristics even when the number of cores is small. The figure shown in the main text is one representative experiment.

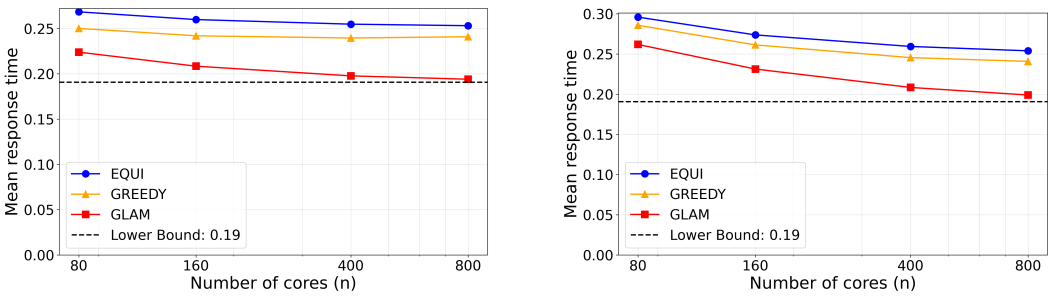
Workload used in the main text. **Class 1:** sub-linear power speedup $s_1(k) = k^{0.9}$, unit mean job size ($\mathbf{E}[X_1] = 1$), weight $c_1 = 1$;
Class 2: Amdahl-law speedup $s_2(k) = \frac{k}{\alpha k + (1-\alpha)}$ with serial fraction $\alpha = 0.2$ (so the asymptotic speedup is capped at $1/\alpha = 5$), unit mean job size ($\mathbf{E}[X_2] = 1$), weight $c_2 = 1$.

Each class carries load $\rho_i = 24$ on a base of 80 cores (so the system load is $\rho = (\rho_1 + \rho_2)/80 = 0.6$). As the number of cores n scales, the arrival rate of each class scales proportionally to keep ρ fixed. The main-text figure (Figure 3) uses Poisson arrivals and Exponential job sizes (SCV = 1).

Additional inter-arrival time and job size distributions. To illustrate that the picture in Figure 3(b) does not depend on the choice of distributions, we repeat the heuristic comparison under two further regimes that fix the workload above but change the variability:

1. **erlang:** Erlang-2 inter-arrival times and Erlang-2 job sizes, a lower-variability regime (SCV = 0.5).
2. **hyperexp:** Balanced two-phase Hyper-Exponential inter-arrival times and job sizes with SCV = 10, a high-variability regime.

The results are shown in Figure K.4. The qualitative behavior matches the exp regime reported in Figure 3(b): GLAM consistently outperforms both heuristics at every n and tracks the lower bound as n grows, while the heuristics do not converge to it.



(a) Erlang (SCV = 0.5)

(b) hyperexponential (SCV = 10)

Figure K.4: GLAM vs. EQUI and GREEDY under inter-arrival and job size distributions with different variability. (a) Erlang-2 inter-arrival times and Erlang-2 job sizes (SCV = 0.5). (b) Balanced two-phase Hyper-Exponential inter-arrival times and job sizes (SCV = 10). In both cases GLAM consistently outperforms EQUI and GREEDY and tracks the lower bound, while the heuristics do not converge to it. System load is fixed at $\rho = 0.6$.